ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE DES TECHNOLOGIES DE L'INFORMATION
M.Sc.A.

PAR
CATUDAL, Serge

MÉTHODOLOGIE PARALLÈLE HYBRIDE APPLIQUÉE AU DÉCODAGE H.264/MPEG-4 AVC SUPPORTANT LA RÉSOLUTION HD SUR UN PROCESSEUR DSP ASYNCHRONE MULTICOEURS

MONTRÉAL, LE 9 AOÛT 2012

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE:

M. Stéphane Coulombe, directeur de mémoire
Département de génie logiciel et des TI à l'École de technologie supérieure

M. Claude Thibeault, président du jury
Département de génie électrique à l'École de technologie supérieure

M. Christian Desrosiers, membre du jury
Département de génie logiciel et des TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 3 AOÛT 2012

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Ce mémoire de maîtrise est le point culminant de plus de trois ans d'études, de recherche et de travail. Je tiens donc à exprimer ma reconnaissance et mes remerciements à Monsieur Stéphane Coulombe, professeur au département de génie logiciel et des TI à l'École de technologie supérieure, qui a accepté d'être mon directeur de maîtrise dans ce projet. Je le remercie aussi pour sa grande disponibilité, ses conseils et son analyse qui m'ont permis de mener à terme mes travaux de recherches.

Je remercie la compagnie Octasic pour m'avoir donné l'opportunité de réaliser ce projet et d'avoir consacré les ressources nécessaires à sa réalisation. Je tiens de plus à remercier mon chef d'équipe, David Bordeleau, de m'avoir fait confiance dans ce projet innovateur.

J'en profite pour remercier tous mes collègues de travail du département de vidéo chez Octasic qui ont participé de près ou de loin à l'élaboration de ce travail, plus particulièrement Denis Leclerc, Louis Poudrier-Racette, Pierre-Yves Duval, ainsi que Daniel Gagné.

Je terminerai en remerciant ma conjointe Sophie Morel qui m'a toujours soutenu tout au long de cette longue aventure.

MÉTHODOLOGIE PARALLÈLE HYBRIDE APPLIQUÉE AU DÉCODAGE H.264/MPEG-4 AVC SUPPORTANT LA RÉSOLUTION HD SUR UN PROCESSEUR DSP ASYNCHRONE MULTICOEURS

CATUDAL, Serge

RÉSUMÉ

Dans le domaine de la téléphonie mobile, la vidéoconférence est une application dont l'adoption est de plus en plus grande. Afin que les fournisseurs de téléphonie mobile puissent rendre accessible une solution unifiée et standardisée de vidéoconférence à un plus grand marché de masse, ceux-ci doivent se tourner vers des solutions plus performantes, flexibles, abordables et consommant moins de puissance. Les nouvelles passerelles média tirent maintenant avantage de processeurs DSP plus performants basés sur des architectures multicoeurs. Pour tirer avantage de ces processeurs DSP, il faut que les implémentations des algorithmes y étant exécutées soient distribuées sur plusieurs coeurs. Un de ces algorithmes est le décodeur provenant de la spécification H.264/MPEG-4 AVC.

Dans ce mémoire, nous proposons une nouvelle solution au décodage en parallèle H.264/MPEG-4 AVC avec profil de base. Cette nouvelle solution tire avantage de l'architecture du processeur DSP asynchrone multicoeurs *OCT1010*. Cette solution se démarque de celles que l'on retrouve dans la littérature principalement parce qu'il s'agit de la première méthodologie parallèle hybride appliquée au décodage vidéo sur un processeur DSP multicoeurs. La solution proposée utilise plusieurs concepts d'extensibilités, plus particulièrement par l'entremise d'un mécanisme d'abstraction de la résolution et par l'entremise d'un mécanisme de synchronisation et d'intercommunication générique extensible en fonction du nombre de coeurs disponibles sur un processeur DSP. De plus, nous proposons, dans ce mémoire, un nouvel algorithme pour améliorer le temps d'exécution de la quatrième étape du décodage de l'entropie de type CAVLC, soit l'extraction du nombre total de zéros à l'intérieur d'un bloc 4x4.

Notre implémentation du décodeur H.264/MPEG-4 AVC, basée sur la solution proposée, a été testée à l'aide de 7 séquences vidéo encodées sous différentes résolutions utilisant différents débits d'encodage. Nos résultats de simulations démontrent que la nouvelle solution au décodage en parallèle H.264/MPEG-4 AVC sur le processeur DSP *OCT1010* atteint les contraintes de *temps réel* pour le domaine de la téléphonie mobile pour les applications de vidéoconférences. En effet, la solution proposée par ce mémoire appliquée sur 11 coeurs DSP pour la résolution HD 720p possède un temps d'exécution représentant 130% des contraintes de *temps réel* comparativement à 18% pour une implémentation séquentielle n'utilisant qu'un seul coeur DSP, ce qui représente un gain d'accélération moyen de 7.3 pour le temps de décodage.

Mot-clés: Décodage parallèle, CAVLC, H.264, vidéo mobile, vidéoconférence

PARALLEL HYBRID METHODOLOGY APPLIED TO H.264/MPEG-4 AVC DECODING SUPPORTING HD RESOLUTION ON AN ASYNCHRONOUS MULTICORE DSP PROCESSOR

CATUDAL, Serge

ABSTRACT

In the field of mobile telephony, the adoption rate of the videoconferencing application is currently growing at a fast pace. In order for mobile phone providers to make a unified and standardized videoconferencing application available to a wider mass market, they must turn to more efficient, flexible, affordable and less power consuming solutions. New media gateways are now taking advantage of more powerful DSP-based multicore architectures. To take advantage of these DSP processors, the implementations of algorithms executed on these processors require to be distributed across multiple cores. One of these algorithms is the decoder from the H.264/MPEG-4 AVC specification.

In this master's thesis, we propose a new solution to parallel decoding applied to the H.264 baseline profile specification. This new solution takes advantage of the *OCT1010* asynchronous multicore DSP processor architecture. This solution differs from those found in the literature mainly because it is the first parallel hybrid methodology applied to video decoding using a multicore DSP processor. The new solution proposed in this master's thesis uses several concepts of extensibility, especially through a resolution abstraction mechanism and on a generic and scalable intercommunication and synchronization mechanism that is based on the number of cores available on a DSP processor. In addition, we propose a new algorithm that improves the execution time of the fourth stage of the CAVLC entropy decoding algorithm which extracts of the total number of zeros inside a 4x4 block.

Our implementation of the H.264/MPEG-4 AVC decoder, based on this solution, was tested using 7 video sequences encoded with different resolutions using different encoding bitrates. Our simulation results demonstrate that our new solution to parallel decoding applied to the H.264/MPEG-4 AVC decoder on an *OCT1010* DSP processor meets *real time* constraints needed in the field of mobile applications for videoconferencing. Indeed, the solution proposed in this master's thesis applied to 11 DSP cores for the 720p HD resolution has a running time representing 130% of *real time* constraints compared to 18% for a sequential implementation using only one single DSP core. This represents a mean decoding time acceleration gain of 7.3.

Keywords: Parallel decoding, CAVLC, H.264, mobile video, videoconference

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES EXTRAITS DE CODE

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ALU	Arithmetic logic unit
AVC	Advanced video coding
CABAC	Context-adaptive binary arithmetic coding
CAVLC	Context-adaptive variable-length coding
CIF	Common intermediate format (352 x 288 pixels)
DSP	Digital signal processor
ETS	École de technologie supérieure
FPS	Frames per second
FIFO	First in, first out
FU-A	Fragmented network abstraction layer unit of type A
GOP	Group of pictures
HD	High definition
IDR	Instant decoder refresh
IP	Internet protocol
MB	Macrobloc
MPEG	Moving Picture Experts Group
NALU	Network abstraction layer unit
NTSC	National Television System Committee (720 x 480 pixels)
QCIF	Quarter common intermediate format (176 x 144 pixels)
RFC	Request for comments

RTP	Real-time transport protocol
SD	Standard definition
SIMD	Single instruction, multiple data
SNALU	Single network abstraction layer unit
TDM	Time-division multiplexing

INTRODUCTION

Problématique

La vidéo est finalement devenue l'application de l'heure que tout le monde attendait (Awad, 2011). Le visionnement de la vidéo sur demande, grâce aux applications internet mobiles, est en forte croissance. De plus, la vidéoconférence est de plus en plus adoptée par les entreprises et les particuliers, notamment grâce à l'émergence des téléphones cellulaires dits *intelligents* et des nouvelles tablettes électroniques. Des applications comme *Facetime*, *Skype*, *GoogleTalk*, et plusieurs autres, peuvent être exécutées autant sur les ordinateurs personnels, que sur les cellulaires ou les tablettes électroniques.

Une des raisons de la plus grande adoption de la vidéo sur les appareils de téléphonie mobile est l'arrivée de résolutions plus grandes, par exemple la résolution HD 720p, offrant une meilleure qualité visuelle à de plus hauts débits. Auparavant, seulement des petites résolutions étaient supportées, comme la résolution CIF ou la résolution QVGA, et cela, à de très bas débits.

Le principal problème des solutions de vidéoconférence actuelles est que ces solutions sont toutes offertes en circuit fermé. Un usager utilisant l'application *FaceTime* ne peut converser avec un usager utilisant l'application *Skype*. Pour unifier et standardiser les communications utilisant la vidéoconférence, les fournisseurs de téléphonie devront utiliser des passerelles média (Arnold et Associates, 2010), aussi appelées *Media Gateway*, supportant des standards vidéos dont notamment le standard provenant de la spécification H.264/MPEG-4 AVC.

La venue de processeurs DSP plus performants et consommant moins de puissance permet maintenant aux équipementiers en télécommunication d'offrir des produits à faibles coûts aux fournisseurs de téléphonie (Awad, 2011). Ces nouveaux processeurs DSP plus performants sont maintenant basés sur des architectures multicoeurs, comme notamment le processeur DSP asynchrone *OCT1010* de la compagnie *Octasic* (Octasic, 2010). Les grands défis dans l'utilisation de tels processeurs DSP reposent sur le fait que chaque coeur DSP, composant le processeur DSP, possède des ressources limitées en puissance de calculs ainsi que des ressources

limitées pour la mémoire locale. Pour tirer avantage de ces processeurs DSP, il faut que les implémentations des algorithmes étant exécutées sur ces processeurs DSP soient distribuées sur plusieurs coeurs DSP.

Un tel algorithme est le décodage vidéo provenant de la spécification H.264/MPEG-4 AVC. Il n'est pas trivial de développer un algorithme de décodage vidéo distribué sur plusieurs coeurs DSP. Un tel algorithme doit répondre à certaines contraintes de performance dont notamment des contraintes de *temps réel*. De plus, un décodeur vidéo n'est pas en contrôle des séquences vidéos qu'il reçoit, c.-à-d. de la manière dont celles-ci sont encodées, ce qui complexifie ses conditions d'utilisation.

Objectifs

Notre effort de recherche vise à étudier les différentes méthodes, pour la parallélisation du décodage, qui sont applicables à la spécification H.264/MPEG-4 AVC et de les combiner pour exploiter l'architecture multicoeurs du processeur DSP asynchrone sur lequel sera implémenté le décodeur H.264/MPEG-4 AVC de ce travail. De plus, nous avons pour objectif d'optimiser le temps d'exécution du décodage de l'entropie de type CAVLC en proposant une solution originale qui améliore le temps d'exécution de l'une de ces étapes.

Notre objectif ultime est de développer un décodeur supportant la spécification H.264/MPEG-4 AVC sur un processeur DSP asynchrone multicoeurs pour le domaine de la téléphonie mobile. Cette implémentation doit supporter des tailles d'images allant jusqu'à la résolution HD 720p tout en respectant une contrainte de *temps réel* pour le débit de trame typique pour les applications de vidéoconférence en télécommunication, qui est généralement de 30 trames par seconde.

Organisation du mémoire

Dans ce mémoire, les informations sont logiquement regroupées en chapitres. La chronologie des chapitres permet de présenter les notions élémentaires en premier, suivies de l'état de l'art,

de la solution proposée et des résultats des simulations servant à mesurer l'efficacité de notre solution.

Le premier chapitre présente les concepts de base concernant les différents aspects et contraintes pour l'implémentation d'un décodeur supportant le standard de compression H.264/MPEG-4 AVC. Dans ce chapitre, nous décrivons le DSP asynchrone multicoeurs sur lequel doit être implémenté le décodeur. Par la suite, nous survolons les différents concepts reliés à ce travail concernant la spécification H.264/MPEG-4 AVC.

Dans le deuxième chapitre, nous effectuons une étude de l'art pour explorer les différentes méthodes pour la parallélisation du décodage qui sont applicables à la spécification H.264/MPEG-4 AVC. Les types de méthodes pour le décodage en parallèle y sont présentés, suivis d'une description plus détaillée des méthodes de parallélisation utilisant la séparation basée sur la fonctionnalité ainsi que la séparation basée sur les données.

Au troisième chapitre, nous présentons la solution proposée au décodage parallèle pour l'implémentation du décodeur H.264/MPEG-4 AVC. Tout d'abord, nous décrivons les types de parallélisation qui sont adoptés et appliqués à notre implémentation. En particulier, nous y présentons une nouvelle solution de décodage parallèle hybride. Par la suite, nous décrivons les concepts d'extensibilités appliquées sur ce même décodeur.

Le quatrième chapitre présente une optimisation effectuée sur le décodage de l'entropie de type CAVLC. Dans ce chapitre, nous décrivons plus en détail les différentes étapes du décodage de l'entropie de type CAVLC. Par la suite, nous présentons la nouvelle solution proposée pour réduire le temps d'exécution de l'une des étapes du décodage de l'entropie de type CAVLC.

Au cinquième chapitre, nous présentons les résultats de nos simulations pour valider l'efficacité des solutions proposées dans ce travail. Au dernier chapitre, nous concluons sur ce travail.

CHAPITRE 1

CONCEPTS DE BASE

Le présent travail a pour objectif principal la réalisation d'un décodeur supportant le standard de compression H.264/MPEG-4 AVC. Ce décodeur doit être implémenté sur un DSP multi-coeur (ou multiprocesseur) asynchrone et doit être dédié principalement pour le domaine de la téléphonie mobile (c.-à-d. le décodage doit se faire en temps réel). Une revue des concepts de base sur les différents aspects entourant cette implémentation devient donc nécessaire pour bien comprendre chacun des aspects concernant ce travail, comme le standard H.264/MPEG-4 AVC et les contraintes d'implémentations sur un réseau mobile.

Ce chapitre est divisé en trois sections principales. À la section 1.1, nous décrivons le DSP asynchrone sur lequel le décodeur H.264 sera implémenté. Par la suite, à la section 1.2, nous décrivons la spécification du décodeur H.264/MPEG-4 AVC. Pour terminer, nous concluons sur les points discutés dans ce chapitre à la section 1.3.

1.1 Description du DSP asynchrone

Le DSP asynchrone d'Octasic, nommé *OCT1010*, est composé de quinze (15) processeurs provenant de la technologie *Opus* (Octasic, 2010). Ces derniers contiennent chacun 16 unités arithmétiques logiques (ALU). Ce DSP consomme très peu de puissance, entre autres grâce à son architecture asynchrone. La plateforme utilisant cette technologie se nomme *Vocallo* et est une passerelle multimédia voix et vidéo (Arnold et Associates, 2010). La plateforme logicielle *Vocallo* est composée d'une interface de communication (*I/O interface device*), d'une mémoire externe (*mobile DDR*), ainsi que d'un processeur DSP asynchrone (*OCT1010 processing device*) (voir la figure 1.1). Cette passerelle est principalement dédiée à la téléphonie mobile sur réseaux IP, donc elle opère sous une contrainte de temps réel, ainsi que sous une contrainte de communication via des paquets Ethernet.

Le DSP *OCT1010* est basé sur une architecture Von Neumann (Null et Lobur, 2006). L'une des principales caractéristiques de cette architecture est que la mémoire locale emmagasine

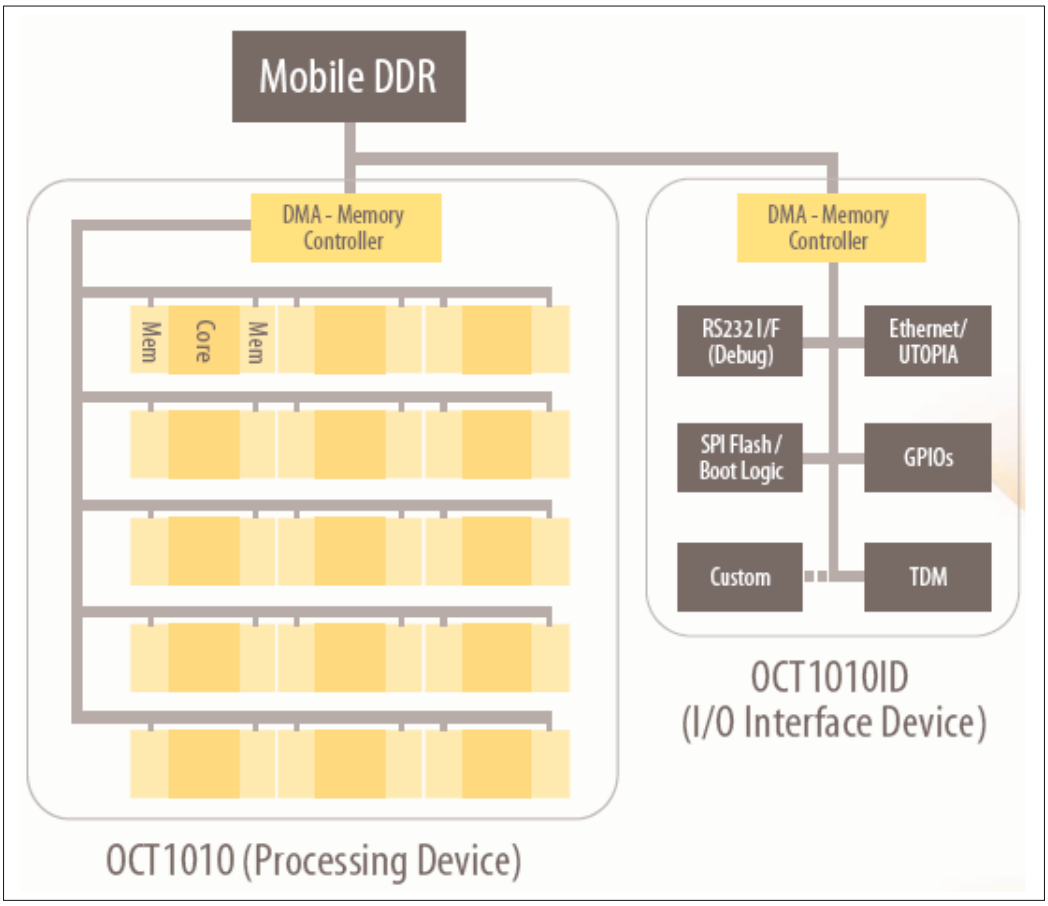


Figure 1.1 Schéma bloc de l'architecture du DSP asynchrone.
Tirée de Octasic (2010)

autant les données que les instructions. Chaque coeur DSP de la technologie *OPUS* possède 96 kilo-octets (Ko) de mémoire locale et ces derniers sont tous reliés à un bus mémoire se nommant *Foundation DMA Bus*. Les différents coeurs DSP peuvent effectuer des accès en mémoire externe via ce bus et peuvent faire des accès internes en écriture à l'intérieur de la mémoire locale d'un autre coeur DSP via ce même bus.

1.2 Le décodeur H.264/MPEG-4 AVC

Le codec provenant de la spécification H.264/MPEG-4 AVC, qui est aussi nommé H.264 tout au long de ce travail, est basé sur les mêmes principes de codage que les standards de compression vidéo précédents. Cependant ce dernier a pour particularités qu'il possède un gain de codage accru grâce, entre autres, à plusieurs nouvelles fonctionnalités et à de nouveaux outils

de compression (Wiegand *et al.*, 2003; ITU, 2007; Richardson, 2003). Ces nouvelles fonctionnalités et ces nouveaux outils de compression apportent évidemment une complexité plus élevée pour le codage et le décodage d'images. Il existe à ce jour dix-huit (18) profils différents pour coder une séquence vidéo avec la norme H.264 et chacun de ces profils possède une liste de fonctionnalités et d'outils de compression qui lui est propre. La figure 1.2 illustre le schéma bloc général du décodeur provenant de la spécification H.264. Ce dernier comporte sept (7) principaux modules de décodage.

Le premier module du standard H.264 est celui du décodage de l'entropie et de l'analyse syntaxique. Ce dernier possède deux nouveaux outils qui se nomment *Context Adaptive Variable Length Coding* (CAVLC) et *Context Adaptive Binary Arithmetic Coding* (CABAC). L'entropie de type CAVLC peut être utilisée par chacun des dix-huit profils. Par contre, l'entropie de type CABAC ne peut pas être utilisée par le profil de base (*baseline profile*), le profil de base restreint (*restricted baseline profile*) ainsi que par le profil Intra 4 :4 :4 (*4 :4 :4 Intra profile*). Comme l'entropie fait partie intégrante du processus de décodage de la syntaxe du standard H.264, ce processus doit alors se faire de manière séquentielle. Cette limitation restreint donc la parallélisation de ce module.

Les deuxième et troisième modules sont la quantification et la transformée inverse. La nouvelle transformée du standard H.264 est généralement faite sur des blocs ayant une taille de 4 pixels par 4 pixels. Il existe une transformée pour des blocs de taille 8 pixels par 8 pixels, mais seulement les profils de type élevé (*high profile*) peuvent employer celle-ci. De plus, cette deuxième transformée doit obligatoirement être combinée au module d'entropie de type CABAC. Toutes les opérations de quantification et de transformée sont faites à l'aide d'une représentation en virgule fixe, ce qui différencie ce standard des autres, car ces derniers utilisent une représentation en virgule flottante.

Le quatrième module est celui du décodage de la prédiction spatiale, aussi nommée prédiction intra. La prédiction spatiale s'opère séparément pour chacune des composantes d'un macrobloc, soit pour les composantes représentant la luminance et représentant la chrominance. Il existe trois types de prédiction spatiale pour la composante de luminance. Le premier type est

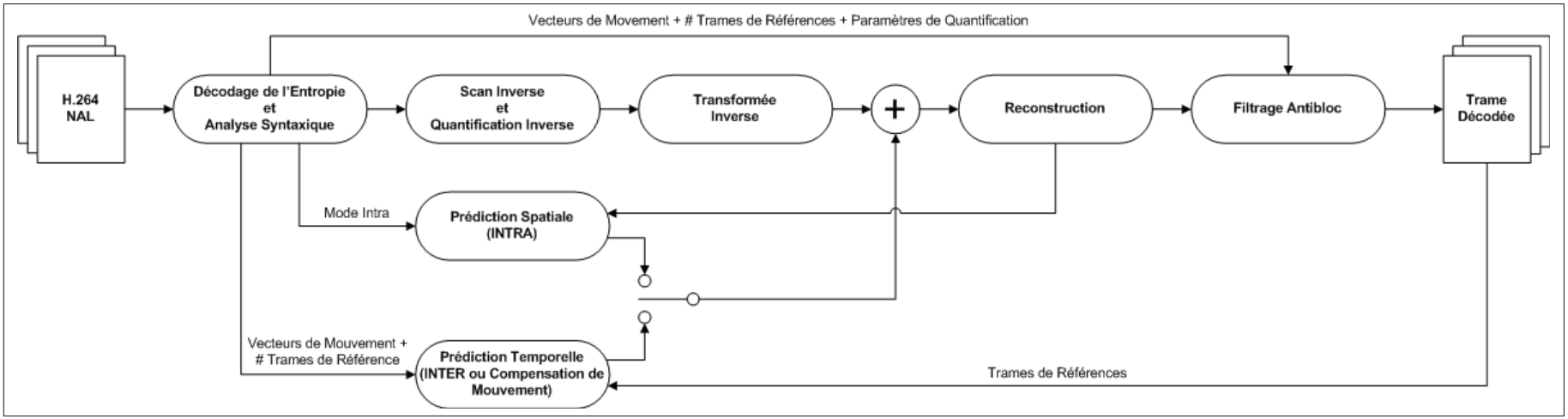


Figure 1.2 Schéma bloc d'un décodeur H.264/MPEG-4 AVC.

la prédiction pour des blocs 4 pixels par 4 pixels, aussi appelée prédiction *intra 4×4*. Celle-ci contient 9 modes (voir la figure 1.4). Le deuxième type est la prédiction pour des blocs 16 pixels par 16 pixels, aussi appelée *intra 16×16*. Cette dernière contient 4 modes (voir la figure 1.3). Le troisième et dernier type est la prédiction pour les blocs 8 pixels par 8 pixels, aussi appelée *intra 8×8*. Ce type de prédiction contient exactement les mêmes 9 modes que la prédiction *intra 4×4*. La prédiction *intra 8×8* est réservée au profil de type élevé (*high profile*) et doit être obligatoirement combinée à la transformée 8 pixels par 8 pixels. La composante de la chrominance utilise une prédiction 8 pixels par 8 pixels. Celle-ci contient les mêmes 4 modes que la prédiction de luminance *intra 16×16*. La prédiction d'un macrobloc (ou bloc) courant se fait à l'aide du macrobloc (ou bloc) voisin de gauche ainsi que du macrobloc (ou bloc) voisin du haut.

Le cinquième module est celui du décodage de la prédiction temporelle, aussi nommé compensation de mouvement. Les figures 1.5 et 1.6 illustrent les types de partitions et de sous-partitions de macroblocs que possède ce module pour le standard H.264. Les standards précédents peuvent utiliser un vecteur de mouvement, et dans certains cas jusqu'à 4 vecteurs de mouvement, alors que le standard H.264 peut utiliser 1, 2, 4, ou 16 vecteurs de mouvement. Ces vecteurs de mouvement sont extraits par le premier module présenté à la figure 1.2, soit celui qui gère le décodage de l'entropie et de la syntaxe. Pour extraire les vecteurs de mouvement, le macrobloc courant dépend du macrobloc voisin de gauche, du macrobloc voisin du haut, ainsi que du macrobloc voisin du haut à droite. Dans certains cas d'exception, soit lorsque le macrobloc voisin du haut à droite n'est pas disponible, l'algorithme d'extraction de vecteurs de mouvement utilise le macrobloc voisin du haut à gauche.

Le sixième module du standard H.264 est celui de la reconstruction et le septième module est celui se nommant filtre antiblocs (*in loop deblocking filter, ou Loop Filter*). Le module de reconstruction utilise les coefficients ayant subi une transformée inverse combinée au résultat de décodage de la prédiction spatiale ou temporelle provenant des troisième, quatrième et cinquième modules présentés à la figure 1.2. Le filtre antiblocs est un filtre servant à enlever les artefacts d'effet de bloc engendrés par la transformée de blocs. Il peut être appliqué de manière

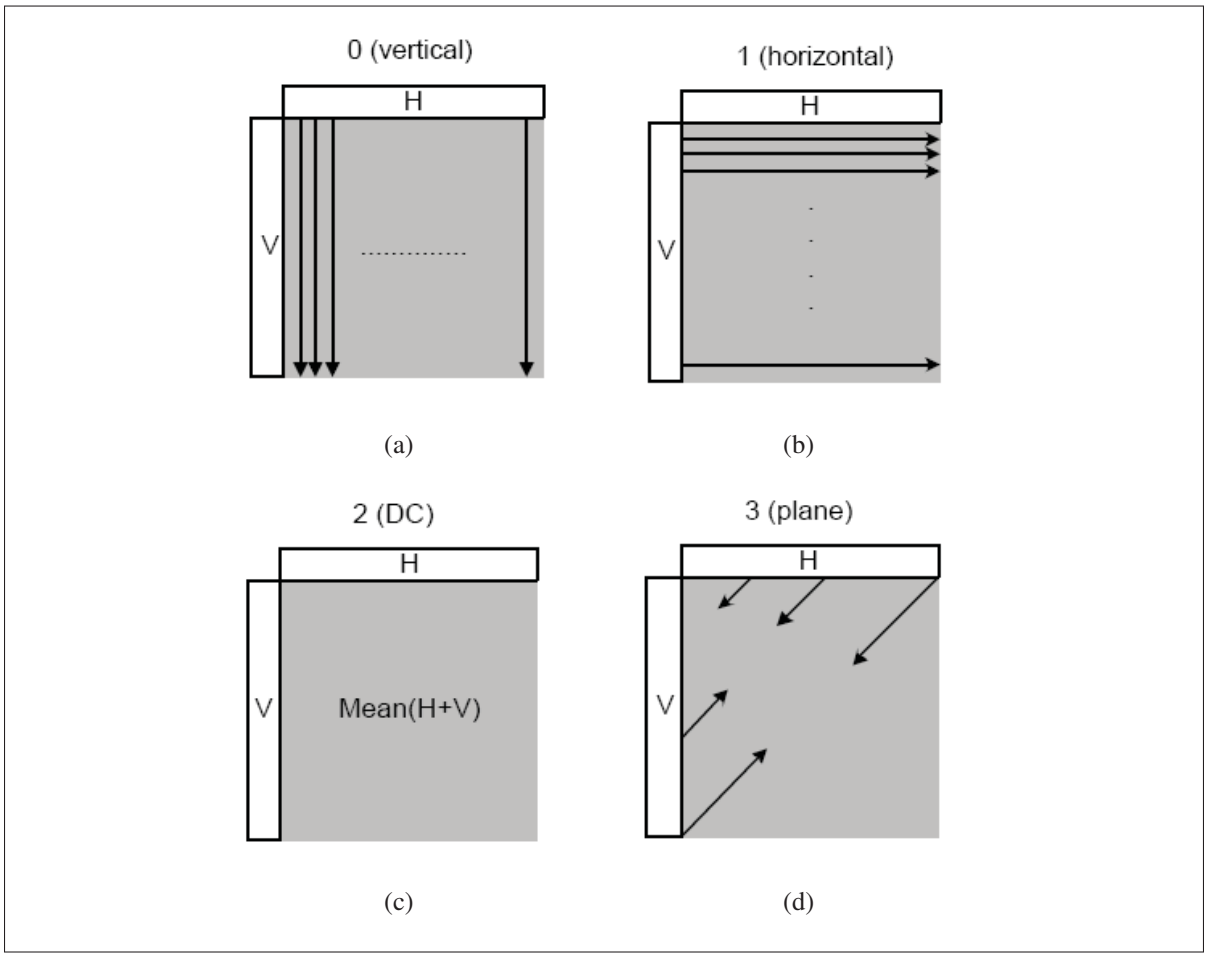


Figure 1.3 Types de prédiction intra 16×16 pour la luminance. Prédiction (a) verticale, (b) horizontale, (c) DC, (d) plan.
Tirée de Richardson (2003)

séquentielle au fur et à mesure que les macroblocs sont décodés, ou il peut être appliqué tout simplement à la fin du décodage de l'image entière. La figure 1.7 illustre la séquence de filtrage définie dans la spécification du codec H.264. Les étapes 1 et 5 représentent un filtrage entre les bordures du macrobloc courant et les bordures des macroblocs voisins de gauche et du haut respectivement, tandis que les étapes 2, 3, 4, 6, 7, et 8 représentent un filtrage entre les blocs internes du macrobloc courant.

Bref, pour effectuer l'étape 1, le macrobloc courant doit s'assurer que le filtrage de son macrobloc voisin de gauche soit complété, alors que pour effectuer l'étape 5, le macrobloc courant doit s'assurer que le filtrage de son macrobloc voisin du haut soit complété. Il faut noter que le

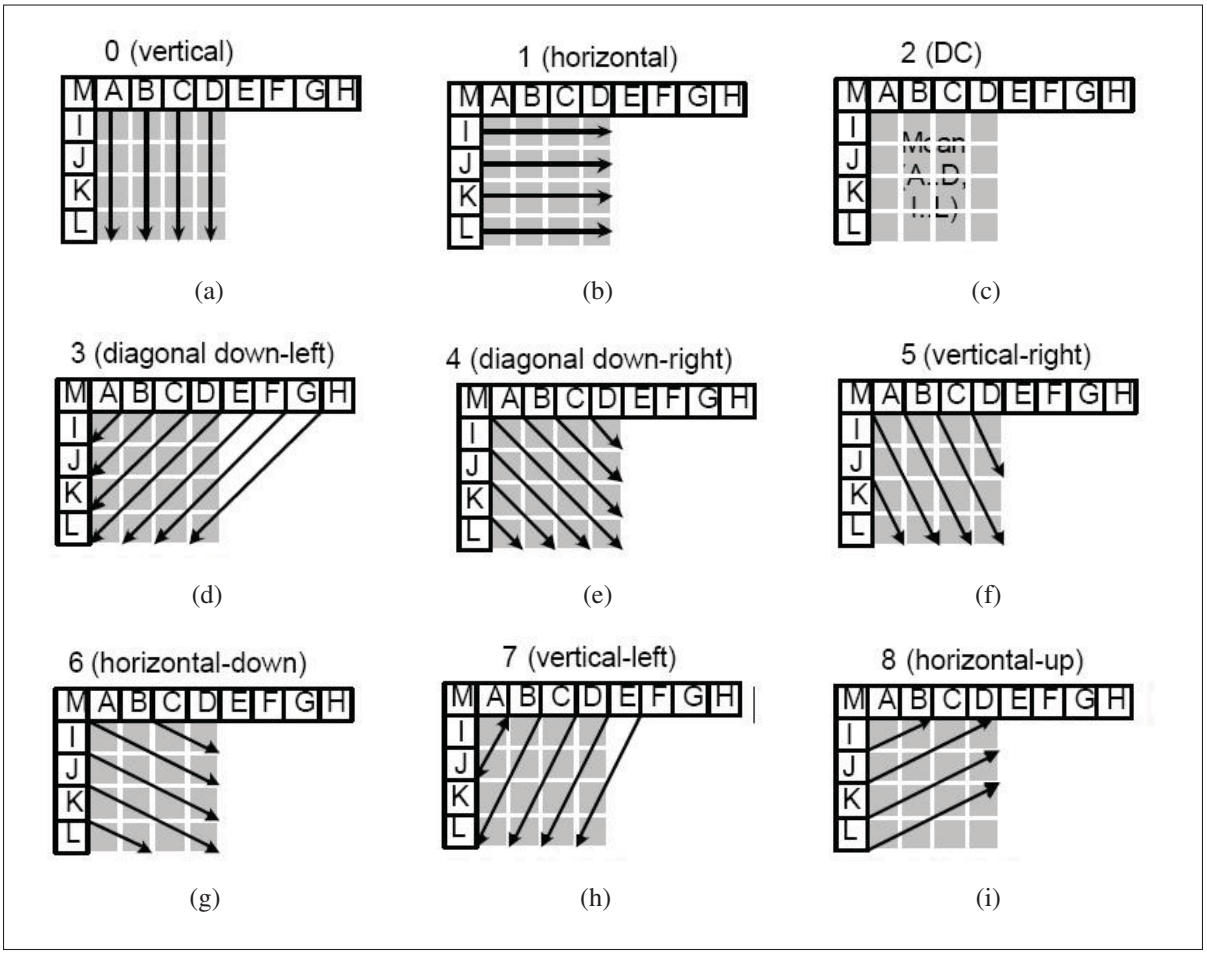


Figure 1.4 Types de prédiction intra 4×4 pour la luminance. Prédiction (a) verticale, (b) horizontale, (c) DC, (d) diagonale bas-gauche, (e) diagonale bas-droit, (f) verticale-droit (g) horizontale-bas, (h) verticale-haut, (i) horizontale-haut.
Tirée de Richardson (2003)

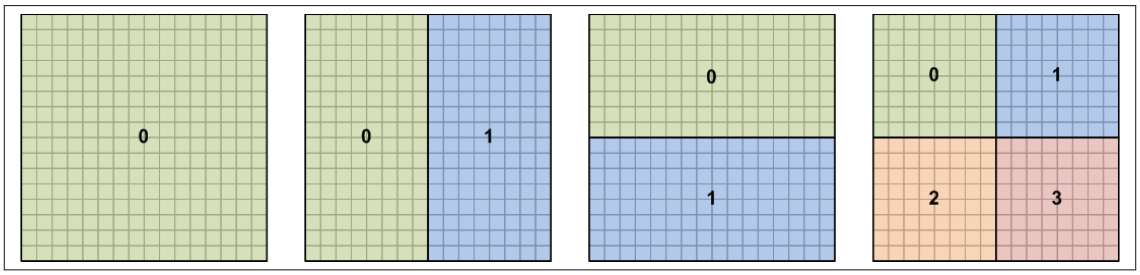


Figure 1.5 Partitions de macrobloc pour la compensation de mouvement : 16×16, 8×16, 16×8, 8×8.

macrobloc voisin du haut est affecté par le filtrage de son macrobloc voisin de droite lorsque ce dernier effectue l'étape 1 du filtrage. Ce macrobloc représente le voisin en haut à droite du

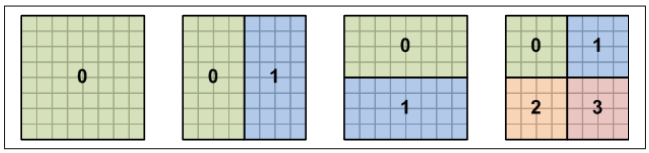


Figure 1.6 Sous-partitions de macrobloc pour la compensation de mouvement : 8×8, 4×8, 8×4, 4×4.

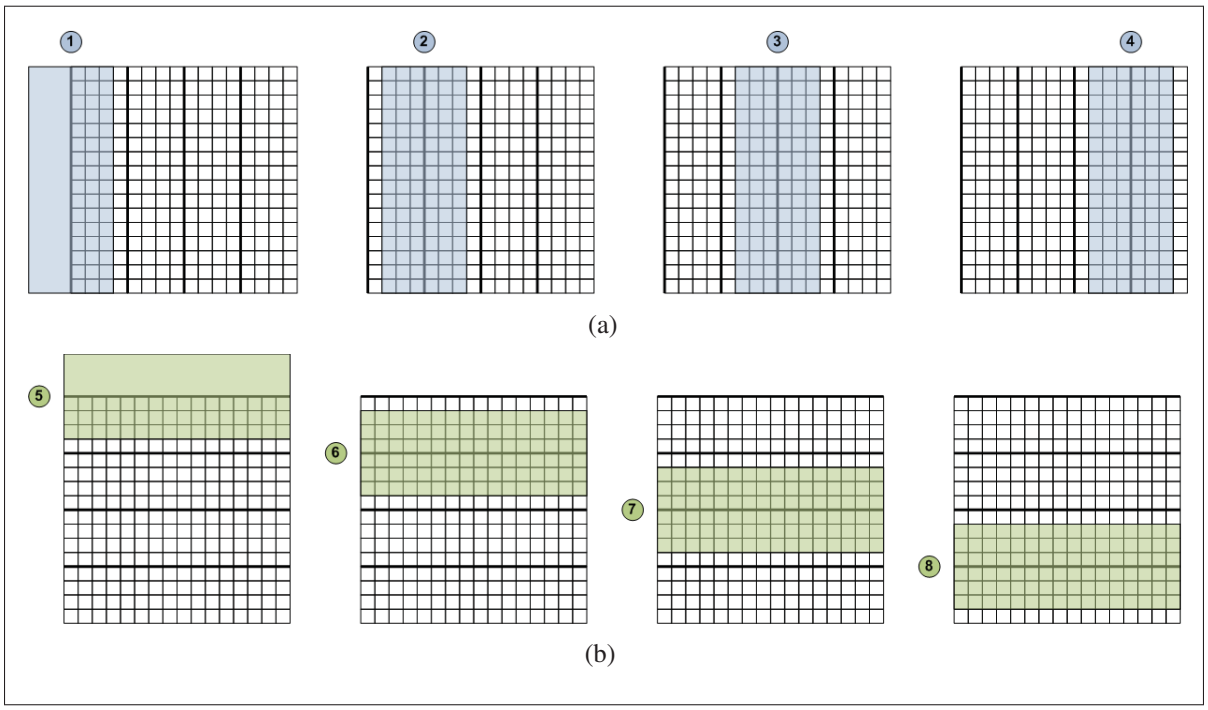


Figure 1.7 Séquence de filtrage de l'effet de bloc (a) verticale, (b) horizontal.

macrobloc courant. Il devient donc une troisième dépendance de macroblocs pour le filtrage de l'effet de bloc, alors que certains articles (Meenderinck *et al.*, 2009) affirment que le module *Loop Filter* ne possède que deux dépendances de macroblocs, soit simplement une dépendance avec le macrobloc voisin du haut et une dépendance avec le macrobloc voisin de gauche.

1.2.1 Structures de données

Il existe plusieurs types de structures de données à l'intérieur de la spécification H.264 (Hui et Hongpeng, 2009; Gurhanli *et al.*, 2011; Meenderinck *et al.*, 2009). La figure 1.8 illustrent les six (6) diverses structures de données composant une séquence vidéo. La première structure de données se nomme groupe de trames (*Group of Pictures*). Un GOP débute obligatoirement par

une trame intra de rafraîchissement instantané du décodage (*Instant Decoding Refresh (IDR) frame*) et est suivi par une ou plusieurs trames inter ou intra. Chaque trame intra ou inter est composée d'une ou plusieurs tranches (*slice*). Chaque tranche d'une trame vidéo est décodable de manière indépendante et est encapsulée à l'intérieur d'une couche d'abstraction de réseau (*Network Abstraction Layer*). Une tranche est composée d'un ou plusieurs macroblocs en fonction de la résolution des trames composant la séquence vidéo. La taille d'un macrobloc pour la spécification H.264 est de 16×16 pixels. Chaque macrobloc est constitué de trois composantes, soit la luminance (Y), la chrominance représentant la différence de couleur bleue (Cb), et la chrominance représentant la différence de couleur rouge (Cr). Un macrobloc est composé de plusieurs blocs de taille 8×8 pixels. La quantité de blocs à l'intérieur d'un MB diffère selon le domaine de couleur utilisé. Il existe trois principaux domaines de couleurs, soit le domaine 4 :4 :4, 4 :2 :2, et 4 :2 :0. Le profil de base de la spécification H.264 utilise le domaine de couleur 4 :2 :0 dans lequel un MB possède quatre (4) blocs de luminance, un (1) bloc de chrominance Cb, et un (1) bloc de chrominance Cr. Chaque bloc est composé de quatre (4) blocs de taille 4×4 pixels. Il sagit de la plus petite structure de donnée à l'intérieur de la spécification H.264.

1.2.2 Dépendances des macroblocs

Comme il fut décrit plus tôt dans ce document, certains algorithmes, ou modules de décodage, du standard H.264 dépendent de données de macroblocs (ou blocs) voisins. La figure 1.9 illustre ces dépendances de données pour un MB courant et ses MB voisins. Dans cette figure, les flèches indiquent que le MB à l'origine de la flèche doit être traité avant le MB courant.

En résumé, l'algorithme du décodage de la prédiction spatiale ainsi que l'algorithme du décodage de l'entropie de type CAVLC possèdent chacun deux dépendances de macroblocs (voir la figure 1.9(a)). Ensuite, l'algorithme de la prédiction des vecteurs de mouvement possède quatre dépendances de macroblocs (voir la figure 1.9(c)). Pour terminer, le module *Loop Filter* possède trois dépendances de macroblocs (voir la figure 1.9(b)). La figure 1.10 résume les différentes dépendances de données entre MB voisins pour les différents algorithmes du standard H.264 (Van Der Tol *et al.*, 2003).

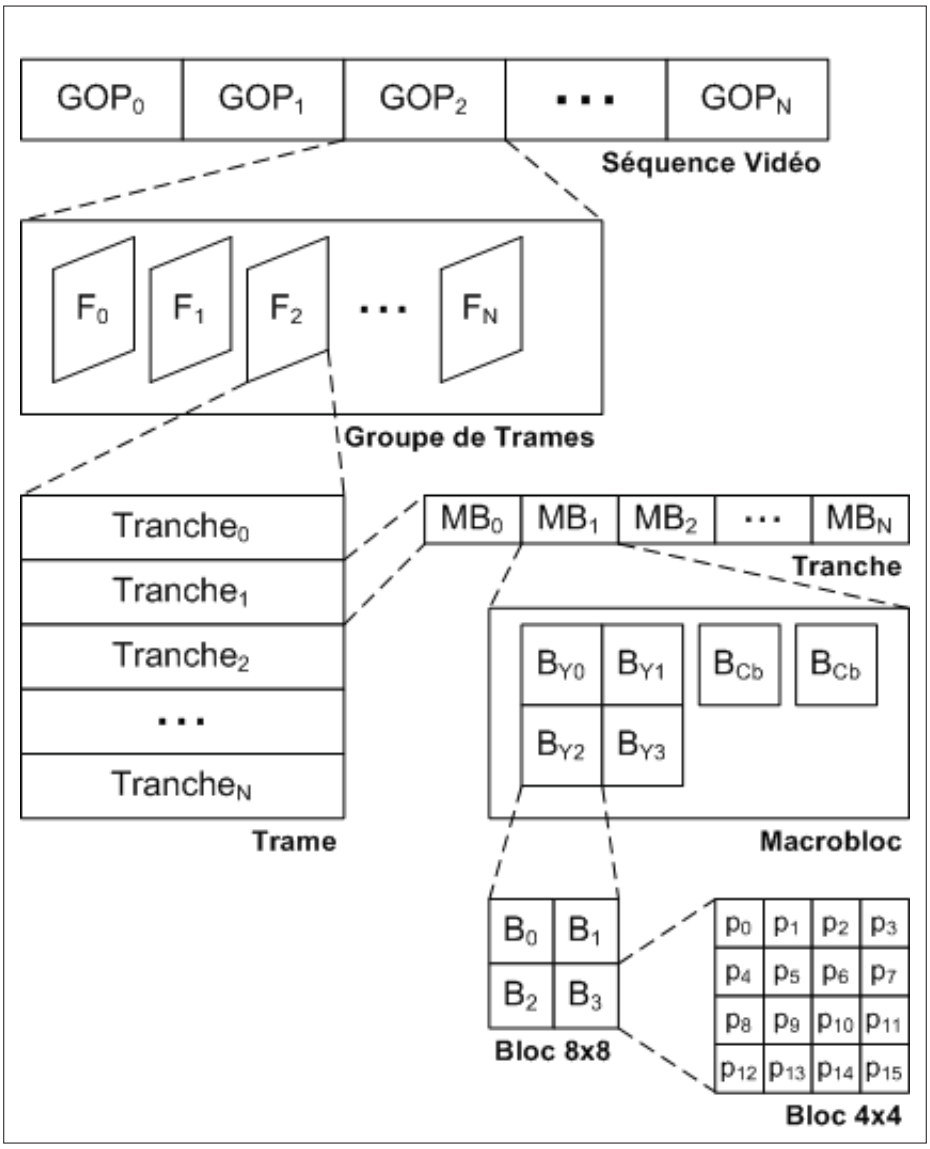


Figure 1.8 Structures de données du décodeur H.264/MPEG-4 AVC.
Adaptée de Gurhanli *et al.* (2011).

1.2.3 Groupage des données par paquets sur un lien en temps réel

Pour transmettre de la vidéo compressée sous le standard H.264 sur un lien IP ou bien sur un lien TDM en temps réel, il faut morceler chaque image en un ou plusieurs paquets. La norme RFC 3984 (Wenger *et al.*, 2005) définit le groupage des données par paquets du codec H.264 sur ce type de lien. Cette norme définit, entre autres, la taille maximale que peut avoir un paquet, son entête, etc. Les deux types de paquet généralement supportés par les appareils de téléphonie vidéo sont le groupage des données par paquets avec tranche unique (SNALU)

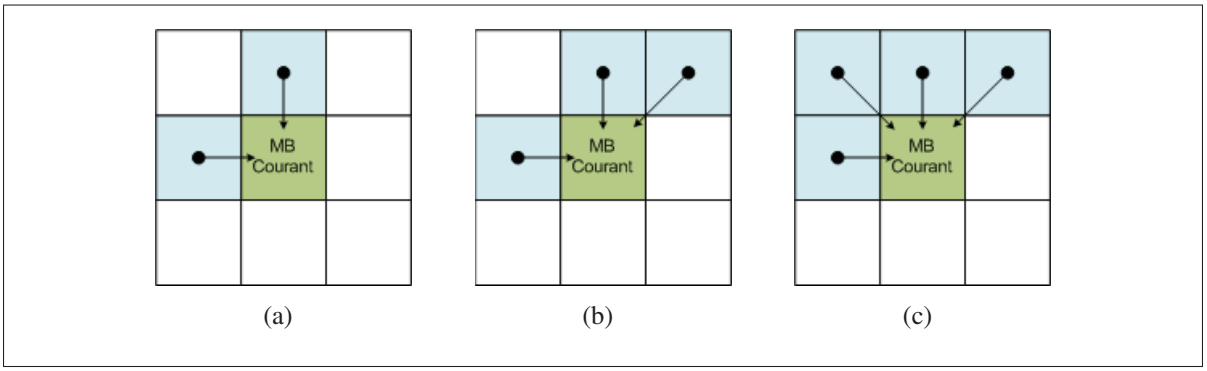


Figure 1.9 Dépendances des macroblocs. (a) MB courant ayant 2 dépendances. (b) MB courant ayant 3 dépendances. (c) MB courant ayant 4 dépendances.
Adaptée de Seitner *et al.* (2011)

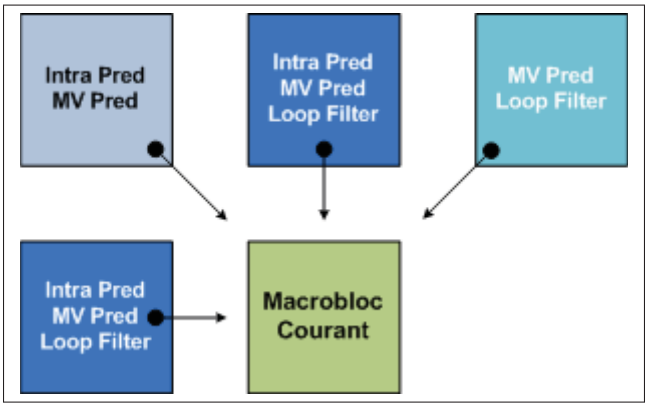


Figure 1.10 Dépendances de données entre des MB voisins pour le codec H.264.
Adaptée de Hui et Hongpeng (2009)

et le groupage des données par paquets avec tranche fragmentée de type A (FU-A). La couche d'abstraction du transport des données pour les images d'une séquence est spécifiquement définie dans le standard H.264.

Le format de transport des données à l'aide de paquets RTP de type SNALU comporte un entête de 1 octet, suivi d'une tranche unique codée à l'aide du standard H.264 (voir la figure 1.11). Comme l'indique clairement son nom, la tranche qui doit faire partie des données de transport du paquet doit entrer en entier dans l'espace maximal réservé à celle-ci.

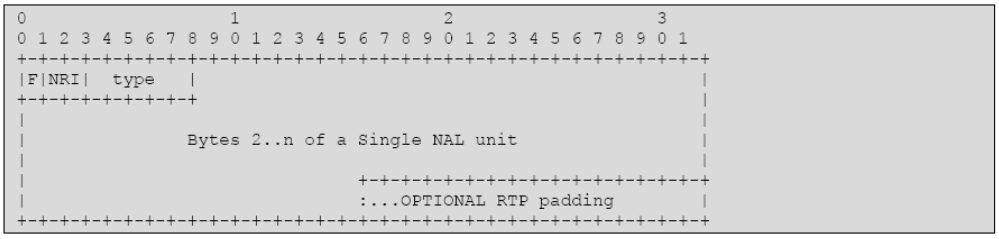


Figure 1.11 Format de transport de données des paquets RTP de type SNALU.
Tiré de Wenger *et al.* (2005)

Si une tranche ne peut pas être contenue en entier dans l'espace qui lui est réservé, il faut alors utiliser le format de transport des données à l'aide des paquets RTP de type FU-A (voir la figure 1.12). Ce format est composé d'un entête de 2 octets, suivi d'un fragment de tranche.

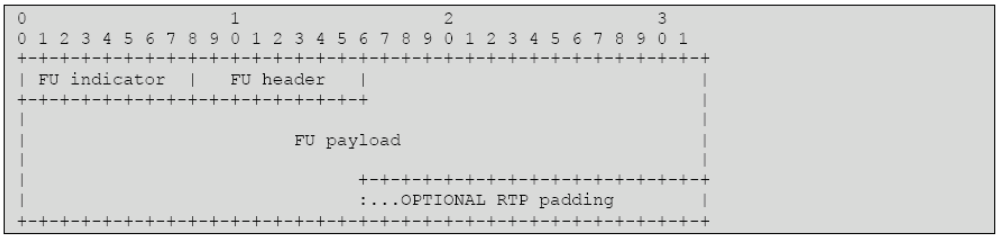


Figure 1.12 Format de transport de données des paquets RTP de type FU-A.
Tiré de Wenger *et al.* (2005)

La figure 1.13 illustre les 2 octets de l'entête des paquets RTP de type FU-A. Dans le premier octet, le membre nommé *Type* doit avoir obligatoirement pour valeur 28. Cela indique qu'il s'agit d'une tranche de type fragmentée. Par la suite, le deuxième octet comporte deux membres très importants. Le membre nommé S est un bit indiquant la présence du premier fragment de la tranche (c.-à-d. lorsque $S = 1$), alors que le membre nommé E est un bit indiquant la présence du dernier fragment de la tranche (c.-à-d. lorsque $E = 1$). Ces deux membres ne doivent jamais être mis à la valeur 1 en même temps, car cela indiquerait que la tranche n'est pas fragmentée (il faut alors utiliser le paquet RTP de type SNALU dans ce cas).

Figure 1.13 Format des 2 premiers octets du transport de données des paquets RTP de type FU-A.
Tiré de Wenger *et al.* (2005)

1.3 Conclusion

Dans ce chapitre, les différents concepts et aspects concernant le décodage avec le standard H.264 sur une plateforme DSP asynchrone multicoeurs dédiée à la classe d'applications de la téléphonie mobile furent élaborés. De plus, chacun des sept (7) modules principaux du décodeur H.264 fut présenté, soit le module du décodage de l'entropie et de l'analyse syntaxique, le module de la quantification, le module de la transformée inverse, le module du décodage de la prédiction spatiale, le module du décodage de la prédiction temporelle, le module de la reconstruction et le module du filtrage antiblocs. Pour terminer, les différentes structures de données composant le décodeur H.264 furent présentées à l'intérieur de ce chapitre ainsi que les différentes dépendances de données entre macroblocs à l'intérieur d'une trame. Dans le prochain chapitre, une étude sur l'état de l'art est effectuée afin d'explorer les différentes méthodes pour la parallélisation du décodage qui sont applicables à la spécification H.264.

CHAPITRE 2

ÉTAT DE L'ART DU DÉCODAGE PARALLÈLE H.264/MPEG-4 AVC

Dans le chapitre précédent, nous avons présenté les concepts de base sur les différents aspects entourant le présent travail, soit l'implémentation d'un décodeur supportant le standard de compression H.264 sur un DSP multicoeur asynchrone pour le domaine de la téléphonie mobile. Le présent chapitre vise à présenter une revue de littérature sur les différents concepts de parallélisation d'algorithmes de décodage dans l'objectif de proposer une solution au décodage parallèle pour l'implémentation du décodeur H.264 avec profil de base pour ce présent travail.

Ce chapitre est divisé en quatre sections principales. À la section 2.1, nous traitons des types de méthodes pour le décodage en parallèle, alors qu'aux sections 2.2 et 2.3 nous décrivons les méthodes de parallélisation utilisant la séparation basée sur la fonctionnalité ainsi que la séparation basée sur les données respectivement. Pour terminer, nous concluons sur les points discutés dans ce chapitre à la section 2.4.

2.1 Type de méthodes pour la parallélisation du décodage

Il existe deux types de méthode pour paralléliser un module de décodage vidéo. Il s'agit de la méthode utilisant une séparation basée sur la fonctionnalité et de la méthode utilisant la séparation basée sur les données (Baik *et al.*, 2007). Celles-ci sont décrites plus en détail dans les sections 2.2 et 2.3 respectivement. À la figure 2.1, deux exemples utilisant quatre (4) processeurs illustrent ces deux méthodes. Chaque couleur à l'intérieur de la figure représente le traitement réalisé par un processeur différent.

La figure2.1(a) illustre un exemple de parallélisation basée sur la fonctionnalité. Dans cet exemple, le premier processeur sert à la synchronisation des trois autres processeurs effectuant respectivement le décodage des coefficients, la prédiction spatiale ou temporelle, et la reconstruction des macroblocs. La figure2.1(b) illustre un exemple de parallélisation basée sur

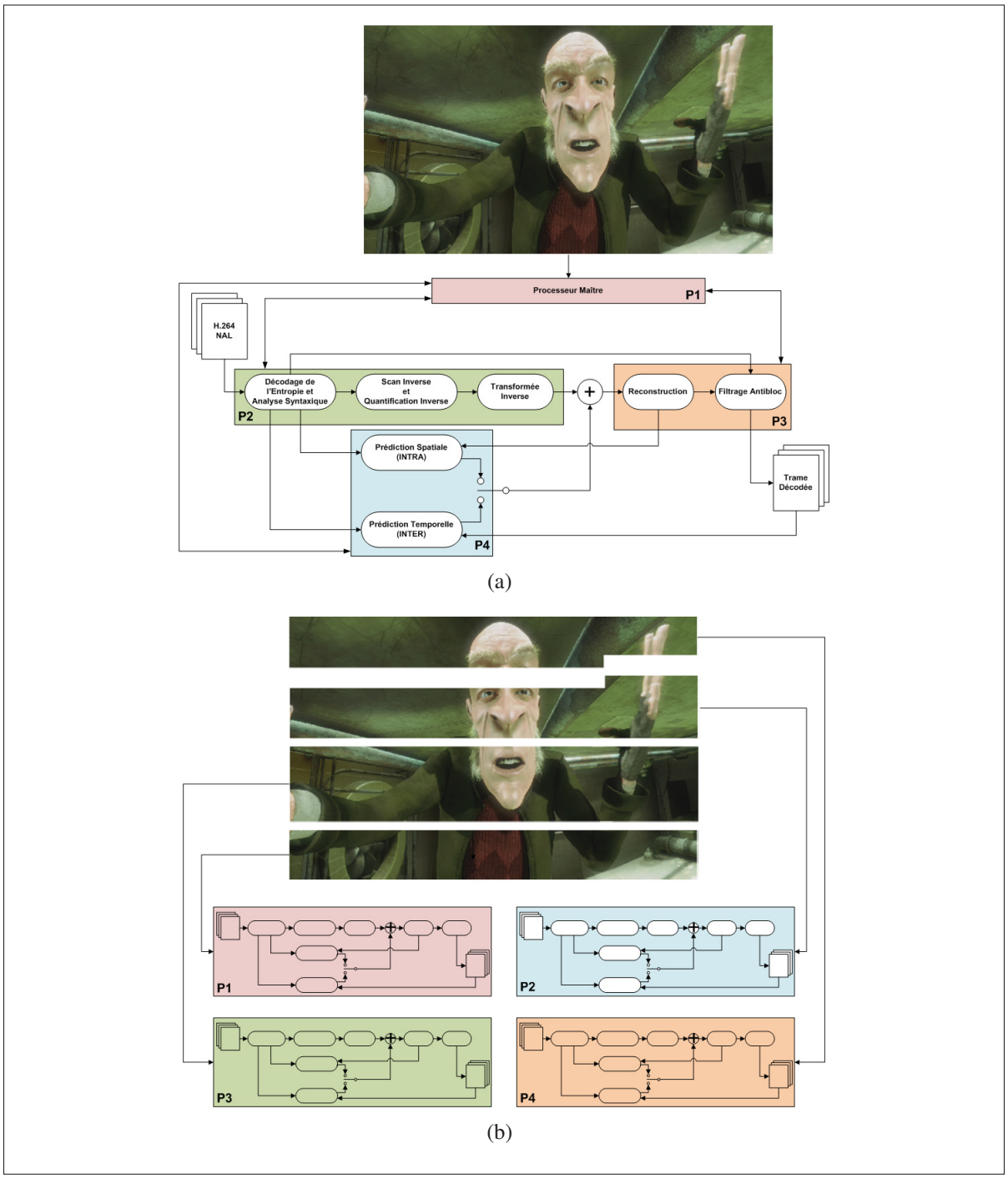


Figure 2.1 Type de méthodes de parallélisation. (a) Basée sur la fonctionnalité. (b) Basée sur les données.

les données. Dans cet exemple, chaque processeur se voit assigner une région de l'image à décoder.

2.2 Séparation basée sur la fonctionnalité

En général, les systèmes utilisant ce type de méthode partitionnent les fonctionnalités liées au décodage, comme la transformée inverse, le décodage syntaxique, la compensation de mouvement, le filtrage de l'effet de bloc, etc., et assignent ces tâches à des processeurs individuels. Typiquement, chaque processeur se voit assigner une ou plusieurs tâches (ou étapes) de décodage (Qiu *et al.*, 2009; Baik *et al.*, 2007). Chaque macrobloc passe donc au travers de chacun des processeurs pour être complètement décodé. Cette décomposition de l'algorithme de décodage en plusieurs niveaux de tâches requiert énormément de communication entre les différents processeurs pour chacune des étapes du décodage. Ce surplus de communication pourrait alors devenir le goulot d'étranglement du processus de décodage. Une solution envisageable pour réduire cet effet non désiré serait d'utiliser une mémoire tampon utilisant un mécanisme de blocage entre chaque processeur, comme par exemple, l'utilisation de FIFO bloquante.

Les principaux désavantages de la séparation basée sur la fonctionnalité concernent l'extensibilité de cette approche ainsi que le balancement du poids d'exécution de chacune des tâches parmi les processeurs disponibles (Meenderinck *et al.*, 2009). En effet, le balancement du poids d'exécution pour chacune des tâches liées au décodage est très difficile à évaluer (ou à prévoir). Le temps d'exécution de chacune des tâches n'est pas connu *a priori* et ce temps d'exécution dépend du contenu des données de chaque image de la séquence vidéo. De plus, au niveau de l'extensibilité de cette approche, si l'application utilise différentes résolutions, le balancement des tâches entre processeurs doit être reconsidéré pour chacune des résolutions.

2.3 Séparation basée sur les données

À l'opposé de la méthode présentée dans la section 2.2, les systèmes favorisant la parallélisation des données ne distribuent pas les fonctions entre différents processeurs, mais distribuent plutôt un ensemble de macroblocs à traiter entièrement. Il est donc plus facile de balancer la tâche de décodage parmi les différents processeurs grâce, entre autres, aux différents niveaux de granularité offerts par cette méthode. De plus, cette méthode est plus extensible pour les systèmes utilisant différentes résolutions d'image. La granularité des méthodes de décodage

en parallèle basée sur la séparation des données sera discutée dans la sous-section 2.3.1. Par la suite, quatre (4) approches pour des systèmes favorisant la parallélisation des données seront présentées et discutées dans les sections 2.3.2, 2.3.3, 2.3.4, et 2.3.5.

2.3.1 Granularité des méthodes de décodage en parallèle

Il existe plusieurs échelons de granularité pour les méthodes de décodage en parallèle dont la séparation est basée sur les données (Meenderinck *et al.*, 2009; Schoffmann *et al.*, 2007). La granularité la plus grossière, soit le premier échelon, est la granularité au niveau d'un groupe d'images (GOP) pour une séquence vidéo. Cette granularité requiert beaucoup de ressources au niveau de la mémoire pour entreposer les images. Cette technique s'applique bien pour les architectures comportant plusieurs ordinateurs. Par contre, celle-ci induit une très grande latence. Elle n'est pas recommandée pour les applications multicoeurs partageant la même ressource mémoire.

Le second échelon est la granularité au niveau des images. Cet échelon peut être utilisé, par exemple, pour paralléliser le décodage de séquences d'images de type I-B-B-P à l'intérieur d'un GOP. Les deux images de type B peuvent être décodées en même temps sur deux processeurs différents. Le seul problème dans cet exemple d'utilisation est que le standard H.264 permet à une image de type B de servir d'image de référence, ce qui n'est pas le cas des standards antérieurs. Il faut donc contraindre l'encodeur à ne pas utiliser les images de type B comme image de référence pour exploiter cet échelon de parallélisme. Cependant, le décodeur n'a aucun contrôle sur l'encodeur en aval, soit celui qui lui fournit la séquence vidéo à décoder.

Le troisième échelon est la granularité au niveau des tranches d'images. En effet, la spécification du standard H.264 permet de partitionner une image en plusieurs tranches indépendantes lors de l'étape de l'encodage. Comme ces tranches sont complètement indépendantes au niveau du décodage, des processeurs en parallèle peuvent décoder chacune des tranches sans contraintes d'ordonnancement. Ceci représente l'avantage principal de cette méthode. Le grand désavantage de cette méthode se situe au niveau du nombre de partitions des tranches dans l'image utilisées par l'encodeur. S'il y a peu de partitions, alors il y aura peu de parallé-

lisation. Il est à noter que le décodeur n'a aucun contrôle sur les partitions de l'image fait par l'encodeur. Dans le pire des cas, il pourrait n'y avoir qu'une seule partition pour l'image en entier.

Le quatrième échelon est la granularité au niveau des macroblocs (Mesa *et al.*, 2009). Cet échelon de granularité peut autant s'appliquer dans le domaine spatial que dans le domaine temporel. Un exemple de l'utilisation de cet échelon de granularité est la technique de l'onde de choc en deux dimensions (2D) présentée dans (Meenderinck *et al.*, 2009). La figure 2.2 illustre cette technique dans le domaine spatial pour une résolution QCIF. Le seul et unique désavantage de cet échelon de granularité est qu'il ne peut pas s'appliquer au décodage de l'entropie, car cette opération doit obligatoirement être effectuée de manière séquentielle.

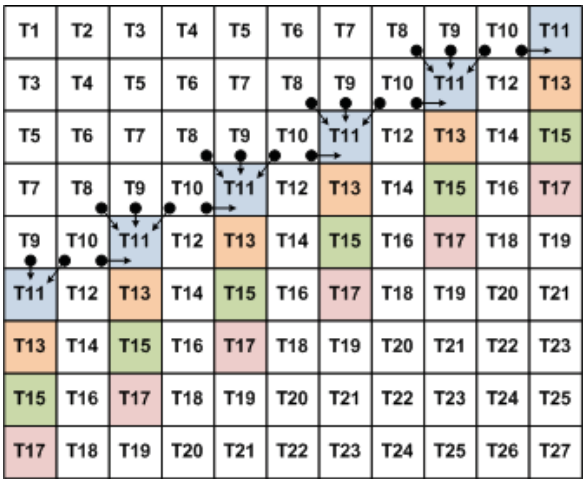


Figure 2.2 Onde de choc en deux dimensions (2D) dans le domaine spatial pour une résolution QCIF.
Adaptée de Meenderinck *et al.* (2009)

La granularité la plus fine, soit le cinquième échelon, est la granularité au niveau des blocs. La plupart des calculs pour le standard H.264 se font au niveau de cet échelon, que ce soit pour la compensation de mouvement, la transformée inverse, ou le filtre antibloc. Cette granularité se prête très bien au parallélisme faisant appel aux instructions de type SIMD (c.-à-d. une simple instruction opérant sur de multiples données). Cet échelon se combine très bien avec l'échelon précédent pour maximiser la parallélisation de la méthode de décodage.

2.3.2 Approche en rangée simple

Cette technique vise à distribuer chaque rangée de macroblocs sur un processeur à la fois (Baker *et al.*, 2009; Chi *et al.*, 2010; Seitner *et al.*, 2008). De façon formelle, le processeur $i \in \{0, \cdots, N-1\}$, où N représente le nombre maximal de processeurs disponibles, se voit assigner la $y^{ième}$ rangée de macroblocs à décoder si $(y \mod N) = i$. La figure 2.3 illustre l'attribution des rangées à leur processeur respectif lorsque deux processeurs sont disponibles (voir la figure 2.3(a)) et lorsque quatre processeurs sont disponibles (voir la figure 2.3(b)).

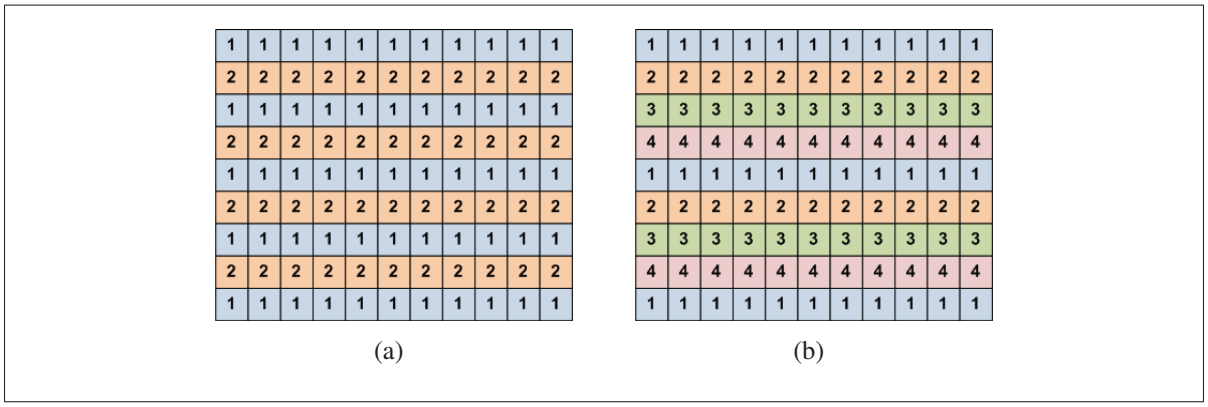


Figure 2.3 Approche en rangée simple utilisant : (a) 2 processeurs, (b) 4 processeurs. Adaptée de Seitner *et al.* (2011)

Pour illustrer le fonctionnement de l'approche en rangée simple, un exemple utilisant deux processeurs est fourni à la figure 2.4. Cet exemple est appliqué sur une image ayant le format QCIF, soit ayant une taille de onze (11) macroblocs horizontaux par neuf (9) macroblocs verticaux. Cet exemple utilise une constante arbitraire de 1 unité de temps d'exécution par macrobloc. À la figure 2.4(a), seul le processeur 1 peut décoder des macroblocs au temps $t = 2$ à cause des dépendances illustrées par la figure 1.10. Lorsque les deux premiers macroblocs de la première rangée sont décodés, le processeur 2 peut commencer à décoder le premier macrobloc de la seconde rangée, car les dépendances entre les macroblocs sont résolues (soit au temps $t = 3$). Le prochain élément intéressant dans cet exemple se déroule au temps $t = 11$, le premier processeur peut commencer à décoder le premier macrobloc de la troisième rangée de l'image. Au temps $t = 13$, le même phénomène qu'au temps $t = 2$ se produit. Finalement, l'image entière

est décodée au temps $t = 55$ (car le processeur 1 commence et termine le traitement et traite séquentiellement 5 lignes à raison de 11 unités de temps par ligne).

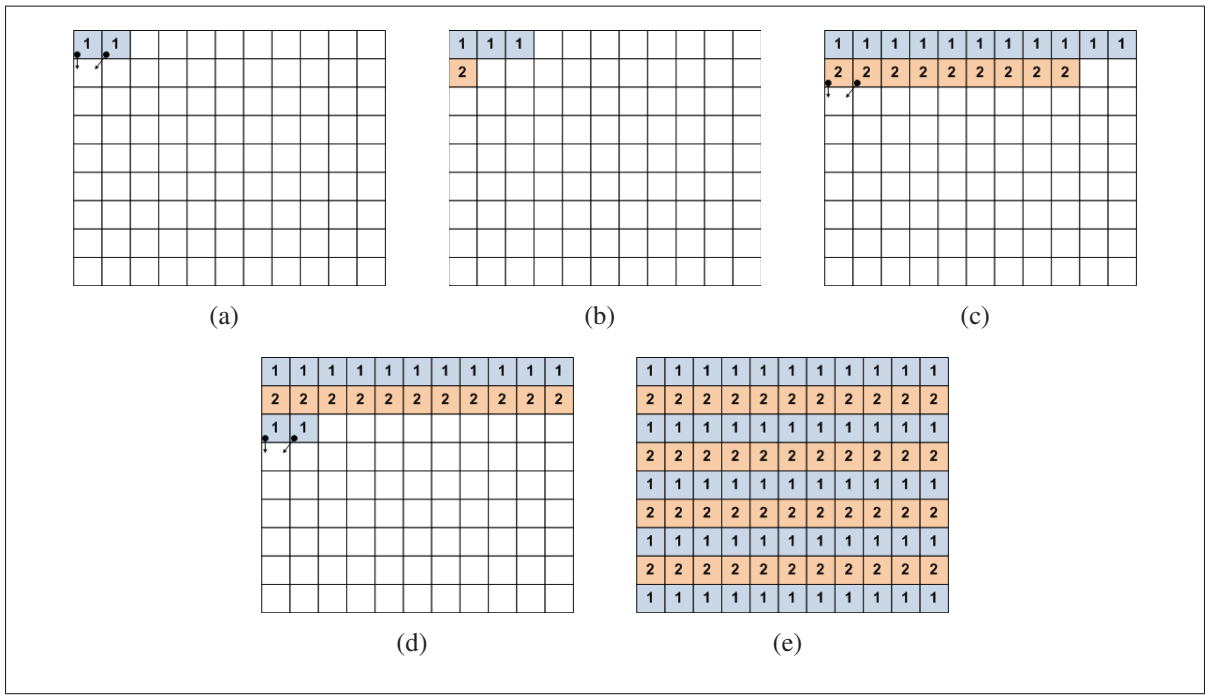


Figure 2.4 Exemple de l'approche en rangée simple utilisant 2 processeurs aux temps : (a) $t = 2$, (b) $t = 3$, (c) $t = 11$, (d) $t = 13$, (e) $t = 55$. Note : le traitement commence au temps $t = 0$.

L'avantage principal de cette solution réside dans sa simplicité. Il est très facile de partitionner une image pour distribuer les opérations de décodage sur différents processeurs lorsqu'il est question de rangées de macroblocs. De plus, la latence (ou le délai) avant qu'un processeur puisse commencer ces opérations est petite. Le grand désavantage de cette solution réside dans le fait que chaque processeur i est dépendant du macrobloc en haut à droite du processeur $i-1$. Si le processeur $i-1$ n'est pas capable de terminer ses opérations sur le macrobloc dont dépend le processeur i, alors ce dernier sera bloqué et inutilisé jusqu'à ce que le processeur $i-1$ complète ses opérations. Selon les résultats d'une étude sur la performance en terme d'images par seconde (Seitner *et al.*, 2011) pour différentes approches de parallélisation du décodage, il a été conclu que les approches utilisant une plus petite granularité et alternant fréquemment les processeurs à l'intérieur de l'image, comme l'approche en rangée simple, s'adaptent beaucoup

mieux aux fluctuations de complexités locales dans une image que les méthodes assignant de façon statique toujours la même région d'une trame au même processeur.

2.3.3 Approche multicolonnes

Cette technique vise à distribuer des colonnes de largeur uniforme de macroblocs sur plusieurs processeurs (Seitner *et al.*, 2011; Sun *et al.*, 2007). De façon formelle, un processeur i est responsable du décodage du macrobloc de la $x^{ième}$ colonne si $(i)w \leq x < (i+1)w$, où w représente la largeur en macrobloc de la colonne. La figure 2.5 illustre l'attribution des groupes de colonnes à leur processeur respectif lorsque deux processeurs sont disponibles (voir la figure 2.5(a)) et lorsque quatre processeurs sont disponibles (voir la figure 2.5(b)).

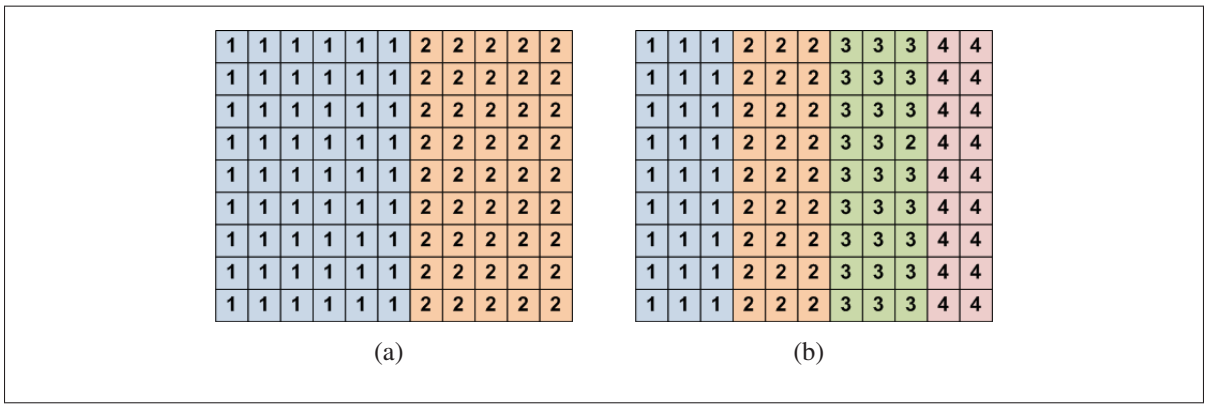


Figure 2.5 Approche multicolonnes utilisant : (a) 2 processeurs, (b) 4 processeurs. Adaptée de Seitner *et al.* (2011)

Pour illustrer le fonctionnement de l'approche en multicolonne, un exemple utilisant deux processeurs est fourni à la figure 2.6. Cet exemple est appliqué sur une image ayant le format QCIF et utilisant une constante arbitraire de 1 unité de temps d'exécution par macrobloc. À la figure 2.6(a), seul le processeur 1 peut décoder des macroblocs au temps $t = 5$ à cause des dépendances de macroblocs. Lorsque le dernier macrobloc de la rangée y du processeur 1 est décodé, le processeur 2 peut commencer à décoder le premier macrobloc de sa rangée y car les dépendances de macroblocs sont résolues (soit au temps $t = 6$). Au temps $t = 12$ et $t = 18$, le même phénomène qu'au temps $t = 5$ se produit. Finalement, l'image entière est décodée au temps $t = 59$.

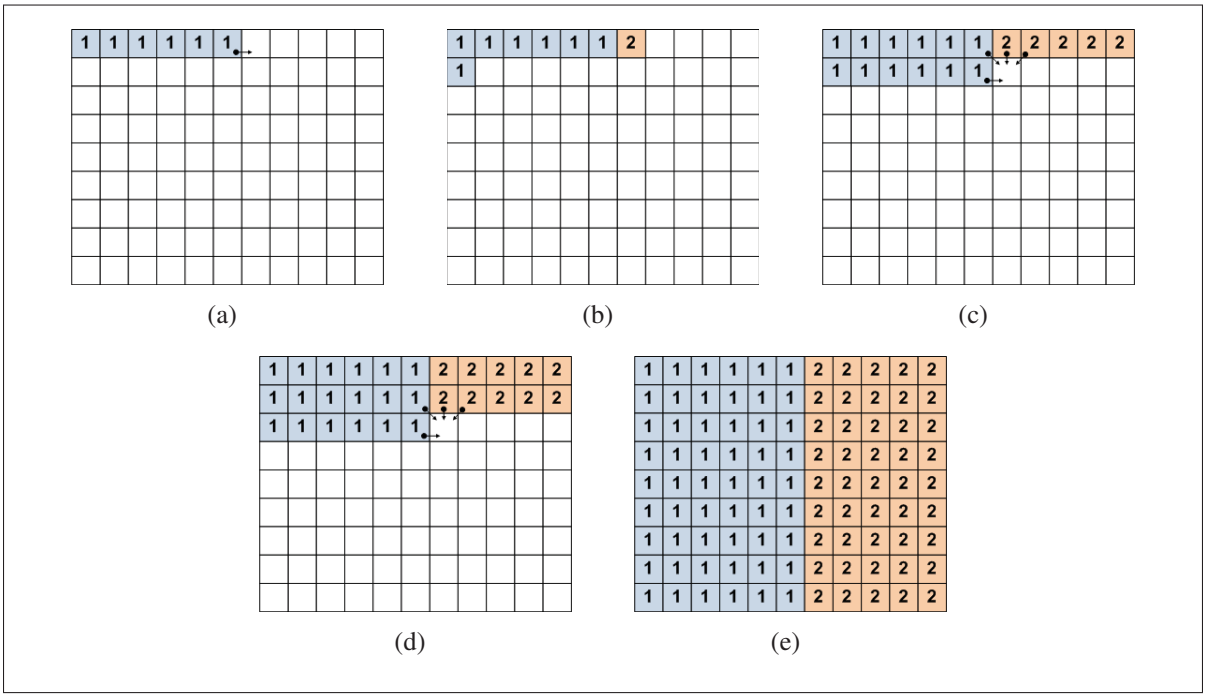


Figure 2.6 Exemple de l'approche multicolonnes utilisant 2 processeurs au temps : (a) $t = 6$, (b) $t = 7$, (c) $t = 12$, (d) $t = 18$, (e) $t = 59$.

Contrairement à l'approche en simple rangée, l'approche multicolonnes a pour avantage que les dépendances de macroblocs entre processeurs se trouvent simplement aux bordures des colonnes (c.-à-d. les bordures de droite et de gauche). Bref, la majorité des dépendances de macroblocs peuvent être résolues par l'entremise du processeur courant, ce qui réduit les dépendances entre processeurs.

2.3.4 Approche multi-tranches

Cette technique s'apparente à l'approche multicolonnes, mais celle-ci est une version de cette dernière soumis à une rotation de 90 degrées (Moriyoshi et Miura, 2008; Seitner *et al.*, 2008). En effet, cette technique vise à distribuer des rangées uniformes de macroblocs sur plusieurs processeurs. De façon formelle, un processeur i est responsable du décodage du macrobloc de la $y^{ième}$ rangée si $(i)w \leq y < (i+1)h$ où h représente la hauteur en macrobloc de la rangée. La figure 2.7 illustre l'attribution des groupes de rangées (ou de tranches) à leur processeur

respectif lorsque deux processeurs sont disponibles (voir la figure 2.7(a)) et lorsque quatre processeurs sont disponibles (voir la figure 2.7(b)).

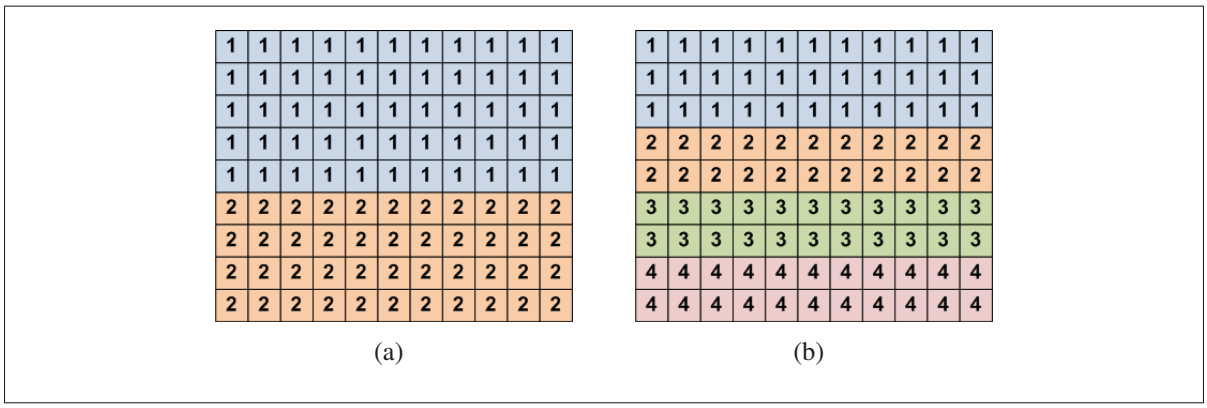


Figure 2.7 Approche multi-tranches utilisant : (a) 2 processeurs, (b) 4 processeurs. Adaptée de Seitner *et al.* (2008)

Pour illustrer le fonctionnement de l'approche multi-tranches, un exemple utilisant deux processeurs est fourni à la figure 2.8. Cet exemple est appliqué sur une image ayant le format QCIF et utilisant une constante arbitraire de 1 unité de temps d'exécution par macrobloc. À la figure 2.8(a), il est possible d'observer que le processeur 2 doit attendre relativement longtemps avant de pouvoir débuter le décodage du premier macrobloc de sa tranche (soit au temps $t = 46$). Le processeur 1 termine le décodage du dernier macrobloc de sa tranche au temps $t = 55$. Finalement, l'image entière est décodée au temps $t = 90$. Il s'écoule donc 36 unités de temps pour que le processeur 2 termine le décodage des macroblocs de sa tranche, alors que le processeur 1 est inutilisé pendant cette période de temps.

Cette technique de parallélisation du décodage est aussi nommée approche multi-tranches bloquante. Il existe aussi une approche multi-tranches non bloquante, cependant cette dernière requiert un contrôle absolu sur l'encodeur en aval du décodeur. Il faut toutefois prendre en considération qu'en présence de cas réels, cette situation ne se présente que très rarement.

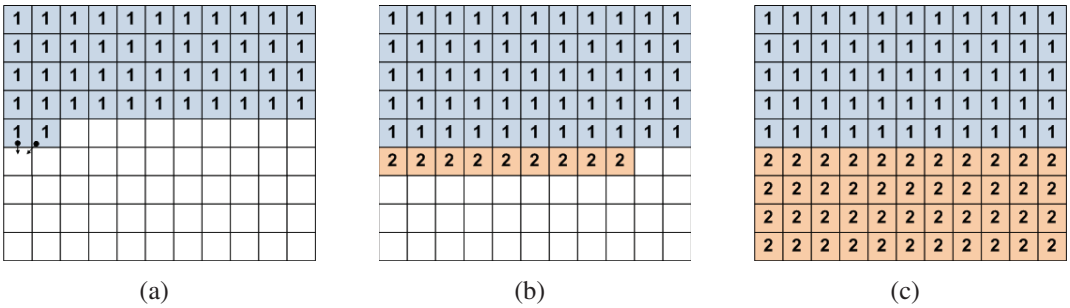


Figure 2.8 Exemple de l'approche multi-tranches utilisant 2 processeurs au temps : (a) $t = 46$, (b) $t = 55$, (c) $t = 90$.

2.3.5 Approche diagonale

Cette technique vise à distribuer des macroblocs alignés en escalier, c.-à-d. une diagonale de macroblocs, sur un processeur à la fois (Seitner *et al.*, 2008; Van Der Tol *et al.*, 2003). L'assignation des processeurs pour la première rangée s'obtient en divisant la première rangée de macroblocs en colonnes de largeurs égales. Les assignations de processeurs sur les rangées suivantes s'obtiennent par le déplacement d'un macrobloc à gauche par rapport à l'assignation de la rangée du dessus. La figure 2.9 illustre l'attribution des groupes de rangées (ou de tranches) à leur processeur respectif lorsque deux processeurs sont disponibles (voir la figure 2.9(a)) et lorsque quatre processeurs sont disponibles (voir la figure 2.9(b)).

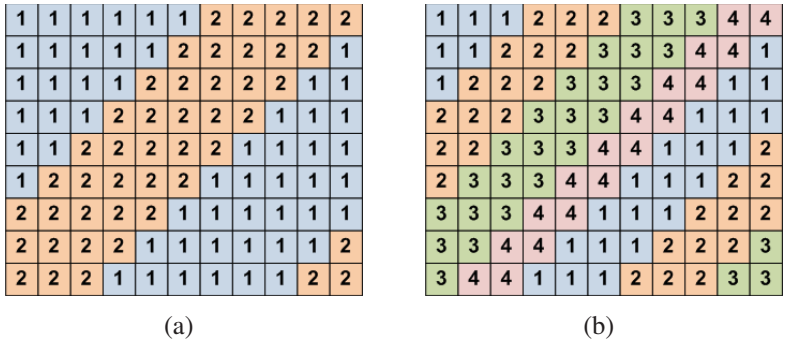


Figure 2.9 Approche diagonale utilisant : (a) 2 processeurs, (b) 4 processeurs. Adaptée de Seitner *et al.* (2008)

Pour illustrer le fonctionnement de l'approche diagonale, un exemple utilisant deux processeurs est fourni à la figure 2.10. Cet exemple est appliqué sur une image ayant le format QCIF et utilisant une constante arbitraire de 1 unité de temps d'exécution par macrobloc. À la figure 2.10(a), seul le processeur 1 peut décoder des macroblocs au temps $t = 5$. Le processeur 2 commence donc son décodage au temps $t = 6$. Le processeur 1 termine sa première diagonale au temps $t = 21$. Le processeur 1 devient plus tard dépendant du processeur 2 pour le décodage de son septième macrobloc de sa nouvelle diagonale au temps $t = 31$ (voir la figure2.10(b)). Les figures 2.10(c) et 2.10(d) illustrent une situation semblable où le processeur 1 est dépendant du processeur 2 pour décoder son macrobloc courant (soit au temps $t = 37$ et $t = 41$ respectivement). Par la suite, la figure 2.10(e) illustre une situation où le processeur 2 est dépendant du processeur 1 pour décoder son macrobloc courant (soit au temps $t = 53$). Une situation semblable se reproduit au temps $t = 59$. Finalement, l'image est entièrement décodée au temps $t = 61$.

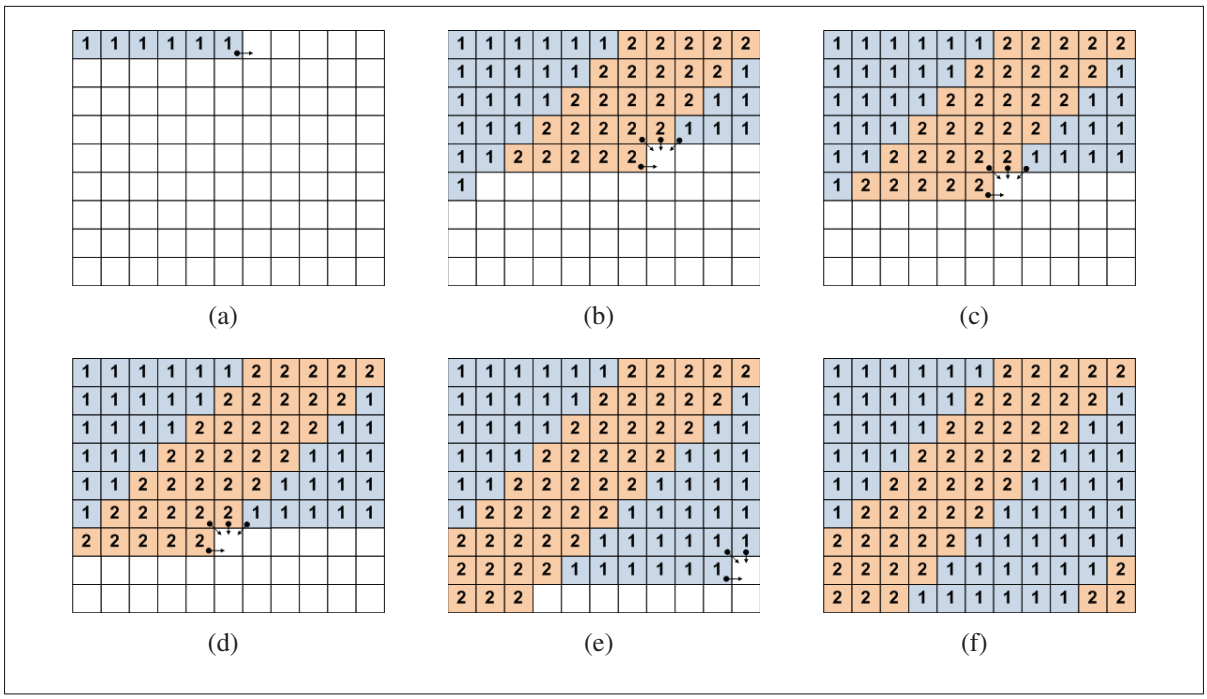


Figure 2.10 Exemple de l'approche diagonale utilisant 2 processeurs au temps : (a) $t = 6$, (b) $t = 31$, (c) $t = 36$, (d) $t = 41$, (e) $t = 53$, (f) $t = 61$.

Avec cette méthode, les dépendances de macroblocs sont présentes seulement avec les macroblocs de gauche du processeur voisin contrairement à la méthode multicolonnes qui possède des dépendances sur les deux côtés de ses frontières. Cette réduction de dépendances entre processeurs pourrait améliorer le temps d'exécution d'un système multicoeurs.

2.3.6 Sommaire sur les approches basées sur les données

Un récapitulation sur les approches basées sur les données est présentée au tableau 2.1. Trois (3) principaux critères pour chacune des approches sont résumés dans ce tableau, soit la latence induite, les types de dépendances de données, ainsi que la taille des transferts de données sur le bus mémoire. Ce dernier critère est très important pour l'implémentation du décodeur H.264 sur un processeur DSP *OCT1010* car ce processeur possède très peu de ressources mémoire interne et tous les transferts de données sont exécutés sur le même bus mémoire (voir la section 1.1 pour la description de l'architecture du DSP asynchrone *OCT1010*). Cela implique que la mémoire locale est très restreinte pour l'emmagasinage de données et qu'un grand nombre de transferts ralentit le système. Pour donner un ordre de grandeur, seulement la résolution QCIF permet de stocker toutes les données nécessaires au décodage et à la reconstruction d'une trame vidéo. Un mécanisme sur l'abstraction de la résolution est présenté au chapitre 3 (section 3.2.1.1) pour pallier cette problématique.

Suite à l'analyse du tableau récapitulatif, le présent travail utilisera l'approche basée sur les données se nommant *rangée simple* pour les processus nécessitant une synchronisation avec des dépendances de données. Le premier motif motivant ce choix est qu'il n'y a seulement qu'un petit délai avant que chaque processeur puisse commencer à travailler. Cette approche maximise donc l'utilisation des différents coeurs DSP. Par la suite, le deuxième motif motivant ce choix est que cette approche permet d'obtenir une taille de transferts de données optimale sur le processeur DSP *OCT1010* avec le gestionnaire de mémoire logiciel utilisé pour l'implémentation du décodeur H.264 de ce travail. La description du modèle du gestionnaire de mémoire logiciel est fournie au chapitre 3 (section 3.1.2) de ce travail.

Tableau 2.1 Récapitulation sur les approches basées sur les données.

Approches	Latence	Dépendances de données	Transferts de données
Rangée simple	Il n'y a seulement qu'un petit délai avant que chaque processeur puisse commencer à travailler, soit le temps de traitement d'un MB.	Il y a des dépendances de données seulement avec le processeur voisin du haut. Par contre, l'exécution du processeur peut-être ralenti lorsque le contenu de sa rangée est moins complexe à décoder que celle de son voisin.	La taille des transferts de données est optimale sur le processeur DSP *OCT1010* avec ce type d'approche.
Multicolonnes	Le délai avant que chaque processeur puisse commencer à travailler est dépendant de la largeur de la trame et du nombre de processeurs utilisés.	Il y a des dépendances de données avec le processeur voisin de gauche et de droite. Ce type d'approche a pour avantage que le processeur peut être ralenti seulement qu'aux frontières entre deux processeurs.	La taille des transferts de données sur le processeur DSP *OCT1010* avec ce type d'approche dépend de la largeur de la trame et du nombre de processeurs utilisés, c.-à-d. qu'elle augmente avec un plus grand nombre de processeurs et avec une trame plus étroite.
Multi-tranches	Il y a un énorme délai avant que chaque processeur puisse commencer à travailler.	Il y a des dépendances de données seulement avec le processeur du haut lors du décodage de la première ligne de chaque processeur.	La taille des transferts de données est optimale sur le processeur DSP *OCT1010* avec ce type d'approche.
Diagonale	Le délai avant que chaque processeur puisse commencer à travailler est dépendant de la largeur de la trame et du nombre de processeurs utilisés.	Il y a des dépendances de données seulement avec le processeur voisin de gauche.	La taille des transferts de données sur le processeur DSP *OCT1010* avec ce type d'approche dépend de la largeur de la trame et du nombre de processeurs utilisés, c.-à-d. qu'elle augmente avec un plus grand nombre de processeurs et avec une trame plus étroite.

2.4 Conclusion

Dans ce chapitre, les deux différentes méthodes pour la parallélisation du décodage avec le standard H.264 furent élaborées. La première méthode est celle fondée sur le partitionnement de l'algorithme à l'aide de la fonctionnalité, tandis que la seconde méthode est celle fondée sur le partitionnement de l'algorithme à l'aide des données. Il existe plusieurs méthodes de partitionnement basées sur les données dans la littérature. Quatre de ces méthodes furent présentées et discutées dans ce chapitre. Il s'agit de l'approche en rangée simple, de l'approche multicolonne, de l'approche multi-tranche, ainsi que de l'approche diagonale. La prochaine étape, présentée au chapitre 3, vise à étudier la stratégie de partitionnement de l'algorithme de décodage H.264 pour la plateforme *Vocallo* pour supporter cet algorithme en temps réel pour des résolutions allant de la résolution standard (SD) à la résolution haute définition (HD).

CHAPITRE 3

SOLUTION PROPOSÉE AU DÉCODAGE PARALLÈLE

Le présent chapitre vise à décrire la solution proposée au décodage parallèle appliqué sur le décodeur H.264 avec profil de base. Cette solution doit répondre à plusieurs critères, dont notamment prendre avantage de l'architecture du processeur spécialisé *OCT1010* comprenant plusieurs coeurs DSP basés sur la technologie OPUS d'Octasic. De plus, cette solution doit être extensible pour n'importe quel type de résolution et doit favoriser un temps d'exécution en *temps réel* pour le domaine de la téléphonie mobile sur le processeur spécialisé *OCT1010* en fonction de la résolution, soit de la résolution QCIF jusqu'à la résolution HD 720p (1280×720 pixels). La méthodologie proposée dans ce chapitre se démarque des solutions se retrouvant dans la littérature parce qu'il s'agit de la première implémentation fonctionnelle d'un décodeur H.264 combinant une approche de décodage parallèle basée sur la fonctionnalité ainsi que sur une approche de décodage parallèle basée sur les données sur un processeur DSP multicoeurs pour le domaine de la téléphonie mobile pour les applications de vidéoconférences.

Ce chapitre est divisé en trois sections principales. À la section 3.1, nous décrivons les types de parallélisation qui sont appliqués pour l'implémentation du décodeur H.264. Par la suite, à la section 3.2, nous décrivons les concepts d'extensibilité appliqués sur ce même décodeur. Pour terminer, nous concluons sur les points discutés dans ce chapitre à la section 3.3.

3.1 Types de parallélisation appliquée au décodeur

L'implémentation du décodeur H.264 proposée dans ce travail utilise une approche de parallélisation hybride (Hui et Hongpeng, 2009), c.-à-d. les trois concepts de parallélisation sont utilisés conjointement. Tout d'abord, le premier concept consiste à utiliser une approche basée sur la fonctionnalité, c.-à-d. le processus du décodage de l'entropie et de l'analyse syntaxique est séparé du processus de la reconstruction et du filtrage antibloc. La figure 3.1 illustre les deux processus utilisés pour cette implémentation du décodeur H.264. Par la suite, le deuxième concept consiste à utiliser une approche de parallélisation basée sur les données pour chacune

de ces deux fonctionnalités. Le troisième concept utilisé pour paralléliser cette architecture est l'ajout du concept de pipelinage (*pipelining*). Le pipelinage est une technique qui consiste à exécuter plusieurs instructions simultanément. Dans la méthodologie proposée, les deux fonctionnalités sont pipelinées. Le schéma de la figure 3.2 illustre les trois concepts utilisés dans l'architecture proposée du décodeur pour paralléliser son exécution.

3.1.1 Processus du décodage de l'entropie et de l'analyse syntaxique

Comme il est précisé dans le chapitre 1, le processus du module de décodage de l'entropie et de l'analyse syntaxique se fait de manière séquentielle. Une trame vidéo encodée à l'aide de la spécification H.264 peut contenir une ou plusieurs tranches (aussi nommées *slice*). Chaque tranche d'une trame vidéo est décodable de manière indépendante. Il s'agit d'un outil fourni par la spécification H.264 servant à rajouter de la résilience aux erreurs à l'intérieur d'une trame vidéo.

Le décodeur ne peut pas prédire le nombre de tranches qui seront présentes à l'intérieur d'une trame vidéo ; ceci est laissé à la libre discrétion de l'encodeur et dépend de son implémentation. Bref, ce processus, soit celui du décodage de l'entropie et de l'analyse syntaxique, est très intimement lié au contenu que le décodeur reçoit et ce dernier ne peut pas contrôler la manière avec laquelle seront encodées les trames vidéos qui lui seront fournies. La figure 3.3 illustre deux trames vidéos encodées à l'aide de 3 et de 6 tranches respectivement.

En ce qui concerne ce processus, le niveau de granularité pour le décodage en parallèle se situe au troisième échelon décrit au chapitre 2 (section 2.3.1), soit au niveau des tranches de macroblocs pour une trame courante. Chaque tranche de macroblocs composant une trame vidéo courante est donc traitée en parallèle et de manière indépendante sur plusieurs coeurs DSP.

3.1.2 Processus de la reconstruction et du filtrage antibloc

Comme il est illustré à la figure 3.1, le processus de la reconstruction et du filtrage antibloc est la fonctionnalité qui regroupe les modules de la quantification, de la transformée inverse,

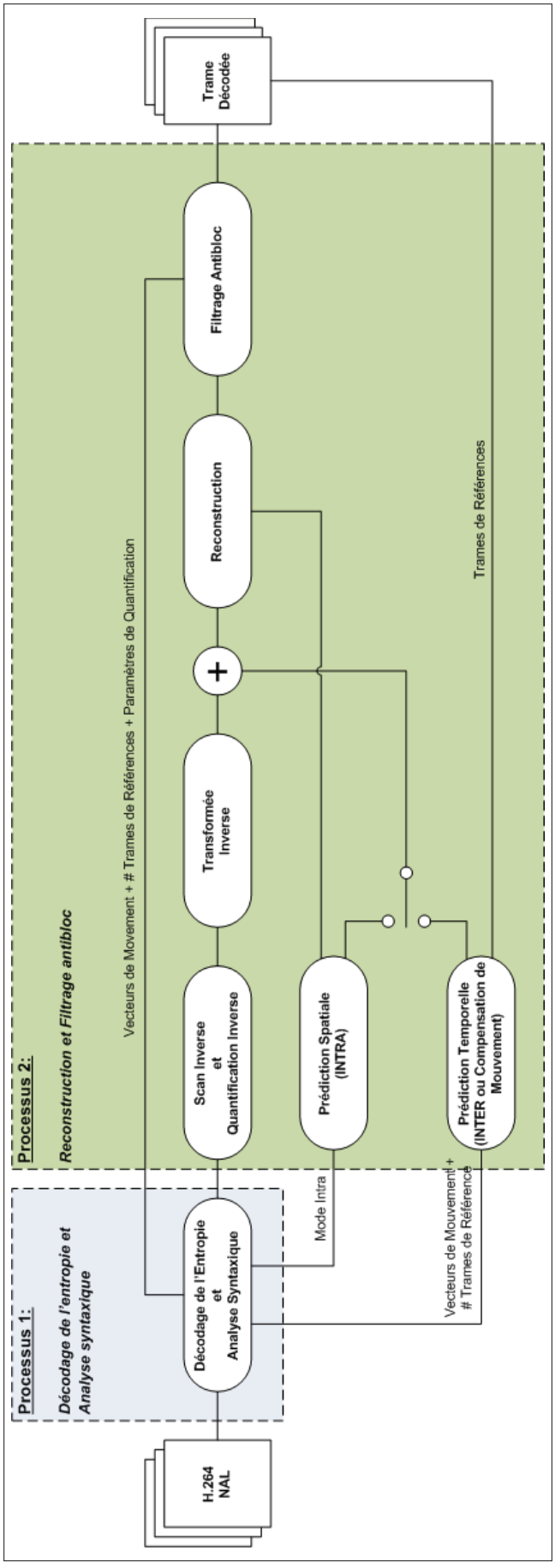


Figure 3.1 Schéma bloc illustrant les deux processus utilisés pour l'implémentation du décodeur H.264.

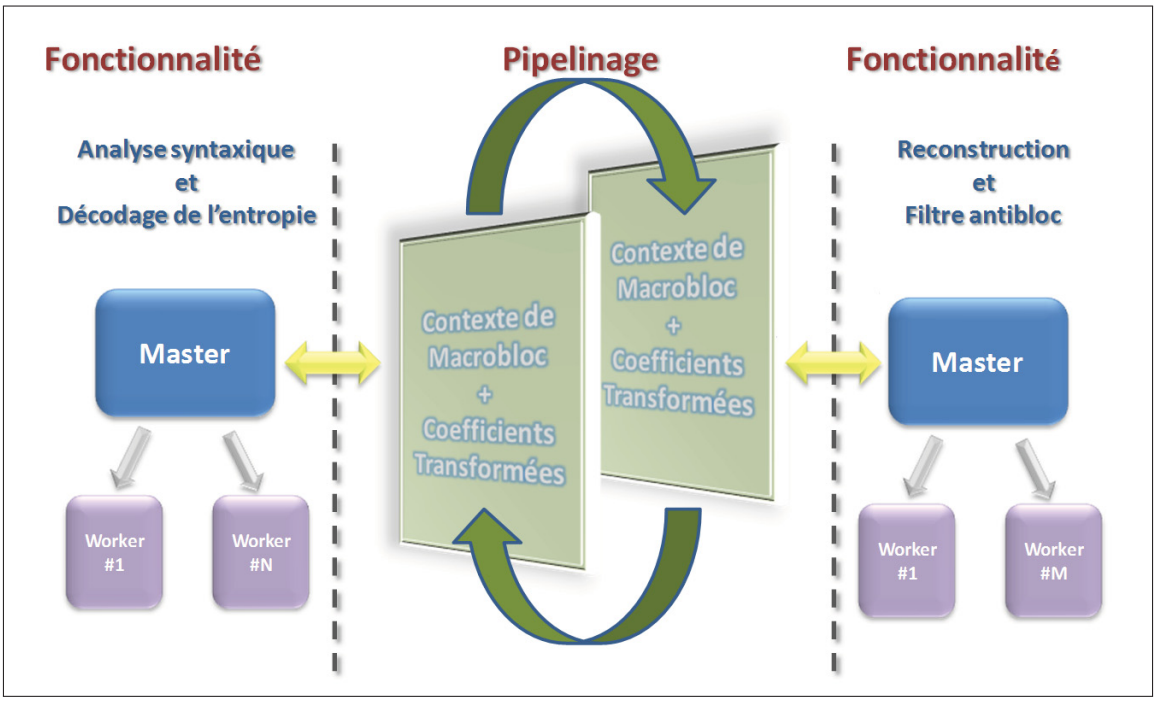
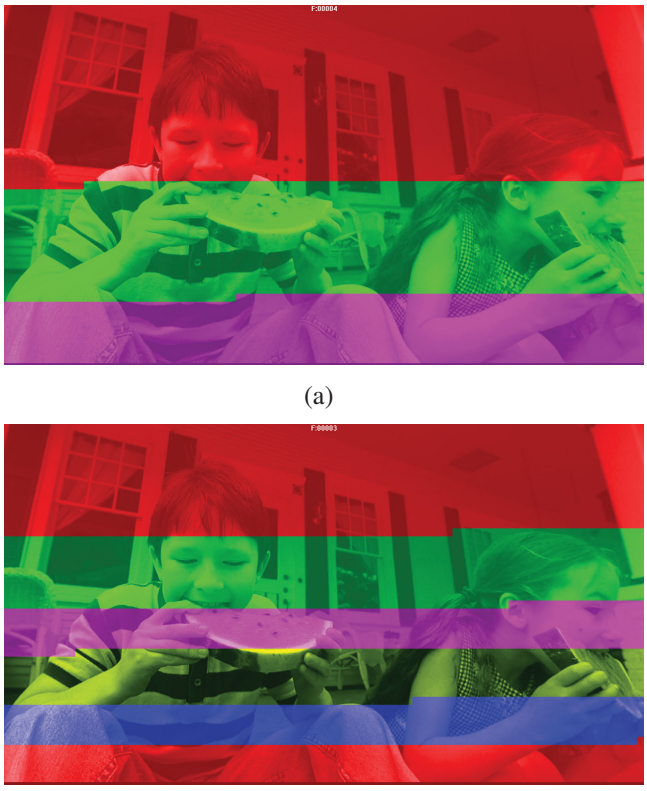


Figure 3.2 Schéma bloc de l'architecture proposée du décodeur.

du décodage de la prédiction spatiale et temporelle, de la reconstruction de macroblocs, ainsi que du module du filtrage antibloc se nommant aussi *Loop Filter*. Chacun de ces modules composant le processus de la reconstruction et du filtrage antibloc comportent des dépendances de données. Ce processus a un niveau de granularité pour le décodage en parallèle se situant au quatrième échelon, soit au niveau des macroblocs. La technique basée sur les données utilisée par ce module, décrite à la sous-section 2.3.2, se nomme *approche en rangée simple*. Dans cette approche, chaque rangée de macroblocs est distribuée sur un coeur DSP à la fois.

L'implémentation du décodeur de ce travail possède un gestionnaire de mémoire logiciel qui abstrait les transferts de données pour le décodeur. En effet, comme les coeurs DSP du processeur *OCT1010* ne possèdent que peu de ressources mémoires locales, le gestionnaire de mémoire s'assure que toutes les données nécessaires pour la reconstruction et le filtrage antibloc pour un MB courant soient présentes localement. C'est ce même gestionnaire de mémoire qui s'assure par la suite de transférer en mémoire externe le résultat de la reconstruction et du filtrage du MB courant. Ce gestionnaire de mémoire, qui abstrait les transferts mémoires pour ce module, est fourni par l'entremise d'une bibliothèque logicielle.



(a)

(b)

Figure 3.3 Exemple de trames vidéos encodées à l'aide de (a) 3 tranches, (b) 6 tranches.

L'approche en rangée simple se prête bien au modèle déjà existant du gestionnaire de mémoire utilisé par le processus de la reconstruction et du filtrage antibloc. Le gestionnaire de mémoire utilise une fenêtre coulissante horizontale asymétrique pour obtenir les données se trouvant en mémoire externe. Cette fenêtre coulissante horizontale sert principalement à l'obtention des données servant à la compensation de mouvement. La figure 3.4 illustre un exemple du déplacement de la fenêtre coulissante sur une rangée de macroblocs. Dans cet exemple, chaque macrobloc rouge représente le MB courant, tandis que les macroblocs jaunes représentent les données disponibles pour le MB courant servant à la compensation de mouvement. Cette zone de macroblocs jaunes est asymétrique.

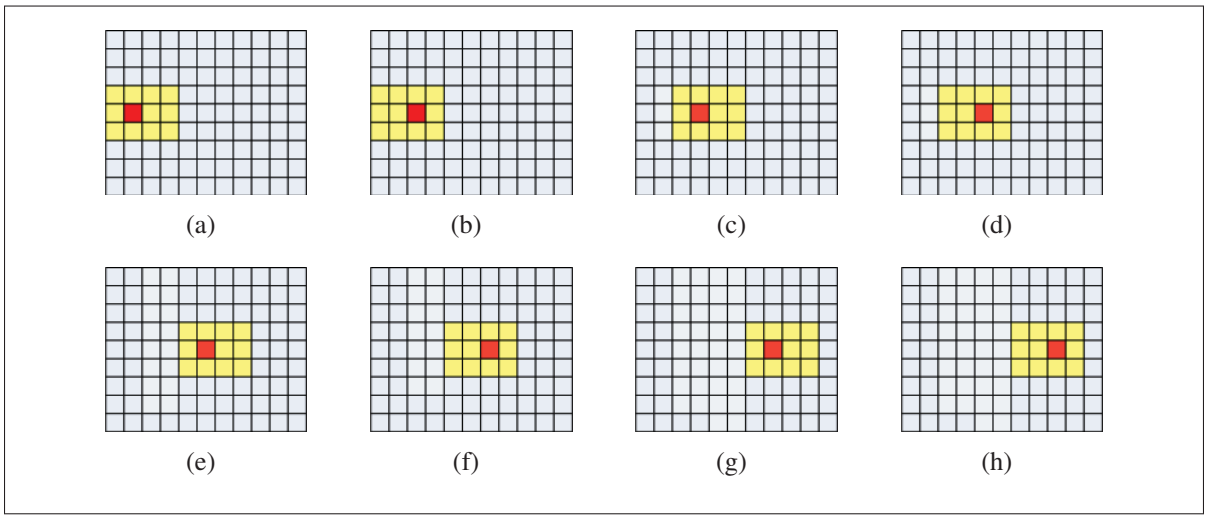


Figure 3.4 Exemple du déplacement de la fenêtre coulissante horizontale sur une rangée de macroblocs.

3.1.3 Pipelinage des deux fonctionnalités

Le débit de trame (aussi appelé *frame rate*) typique pour les applications de vidéoconférence en télécommunication est généralement de 30 trames par seconde (30fps). À ce débit, le décodeur doit pouvoir décoder une trame en 33 millisecondes. Le processus du décodage de l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc ont donc un temps de traitement combiné de 33ms pour répondre à cette contrainte de temps réel. En pipelinant les deux fonctionnalités (c.-à-d. le processus du décodage de l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc), le décodeur H.264 obtient alors une meilleure flexibilité et une plus grande marge de manoeuvre pour décoder les trames vidéos en temps réel. En effet, chaque fonctionnalité possède dorénavant un temps maximal de traitement individuel de 33ms. Le mécanisme de pipelinage entre les deux fonctionnalités est expliqué dans le paragraphe suivant.

Pour implémenter ce mécanisme, les deux fonctionnalités partagent une mémoire tampon, ayant une profondeur de deux trames, dans laquelle se retrouvent les contextes de macroblocs et les coefficients transformés pour une trame (voir la figure 3.2). Aussitôt qu'une fonctionnalité a terminé de traiter les données avec le tampon d'une trame, l'autre fonctionnalité peut

l'utiliser dès qu'elle a terminé le traitement des données sur le tampon précédent. La figure 3.5 illustre un exemple de l'utilisation d'une mémoire tampon ayant une profondeur de deux trames. Par exemple, si la fréquence de trame est de $f_t = 30fps$, alors le délai maximal est de $\Delta_t = 2 \times (1/f_t)$ plutôt que $\Delta_t = (1/f_t)$.

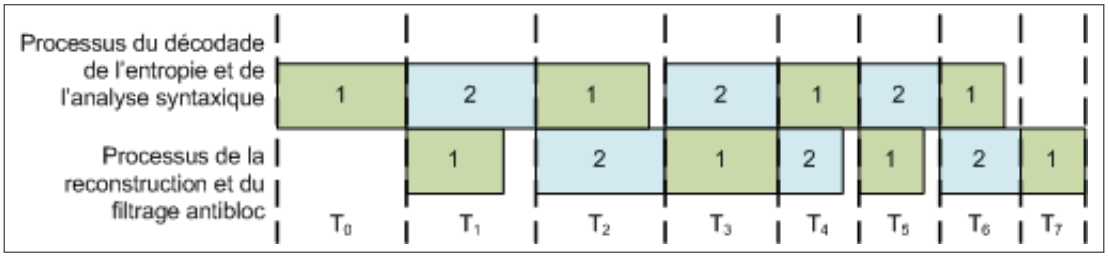


Figure 3.5 Exemple de l'utilisation de la mémoire tampon entre les deux fonctionnalités.

3.2 Extensibilité du décodeur

Comme mentionné précédemment, l'implémentation du décodeur H.264 se fait sur un processeur spécialisé comprenant plusieurs coeurs DSP ayant des ressources mémoires locales très limitées. De plus, cette implémentation doit répondre à un critère principal qui est de décoder des séquences d'images avec diverses résolutions pouvant aller jusqu'à la résolution HD 720p en temps réel. En prenant en compte ces diverses caractéristiques, le concept d'extensibilité devient alors très important pour l'implémentation du décodeur pour obtenir une solution générique en termes de ressource mémoire et de nombre de coeurs DSP disponibles. La sous-section 3.2.1 décrit la méthode d'extensibilité appliquée au modèle mémoire, alors que la sous-section 3.2.2 décrit la méthode d'extensibilité appliquée au modèle de synchronisation et d'intercommunication afin que le décodage soit indépendant du nombre de coeurs DSP.

3.2.1 Modèle mémoire

Comme il est mentionné précédemment, le processeur spécialisé sur lequel est implémenté le décodeur H.264 possède des ressources mémoires locales extrêmement limitées. En effet, la mémoire cache locale (niveau L1) de chacun des coeurs DSP est de 96 kilo-octets (ko) et cet espace mémoire sert autant pour emmagasiner le code devant être exécuté sur ce coeur DSP que pour l'emmagasinage des données. De plus, 24 ko sont réservés exclusivement pour le

noyau du système d'exploitation, ce qui représente une perte d'espace de 25%. Il reste donc 72 ko disponibles pour le code et les données servant au décodeur H.264, ce qui implique qu'il faille minimiser les données se trouvant localement sur le coeur DSP, et cela, indépendamment de la résolution de l'image étant décodée. Pour fournir un ordre de grandeur, une trame vidéo non compressée ayant la résolution HD 720p requiert à elle seule 1350 ko.

3.2.1.1 Mécanisme d'abstraction de la résolution

Pour réduire la taille des données présentes localement sur un coeur DSP, le décodeur utilise un mécanisme qui abstrait la résolution d'une trame vidéo. Ce mécanisme vise à transférer le minimum de données nécessaires entre la mémoire locale et la mémoire externe pour prendre en considération les dépendances de données se trouvant entre des macroblocs (voir la section 1.2.2 du chapitre 1 pour un rappel sur les dépendances de données entre des macroblocs). Ce mécanisme d'abstraction de la résolution utilise cinq (5) conteneurs génériques représentant des données pour le macrobloc voisin du haut à gauche (*Top-Left MB*), le macrobloc voisin du haut (*Top MB*), le macrobloc voisin du haut à droite (*Top-Right MB*), le macrobloc voisin de gauche (*Left MB*), ainsi que le macrobloc courant (*Current MB*). La figure 3.6 illustre les cinq conteneurs génériques utilisés en mémoire locale pour représenter des macroblocs voisins.

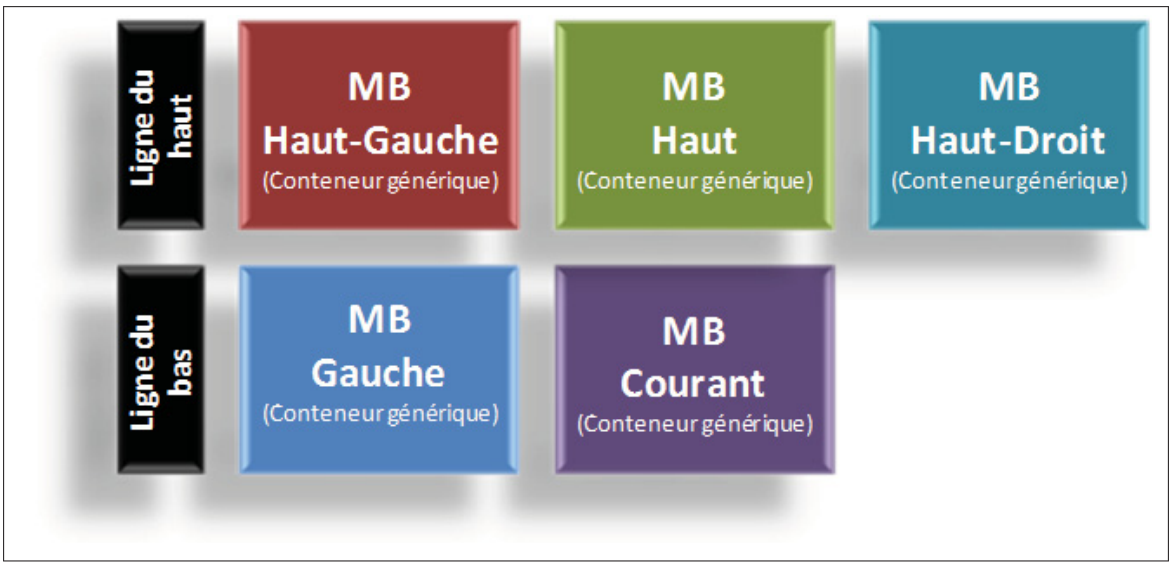


Figure 3.6 Les 5 conteneurs génériques utilisés pour représenter des macroblocs voisins.

La figure 3.7 illustre un schéma haut niveau du système se trouvant sur le processeur DSP *OCT1010*. Dans ce système, trois coeurs DSP sont réservés exclusivement aux tâches de contrôle, de routage et d'ordonnancement. Tous les autres coeurs DSP sont disponibles afin d'exécuter les autres tâches du système, par exemple, décoder une séquence vidéo.

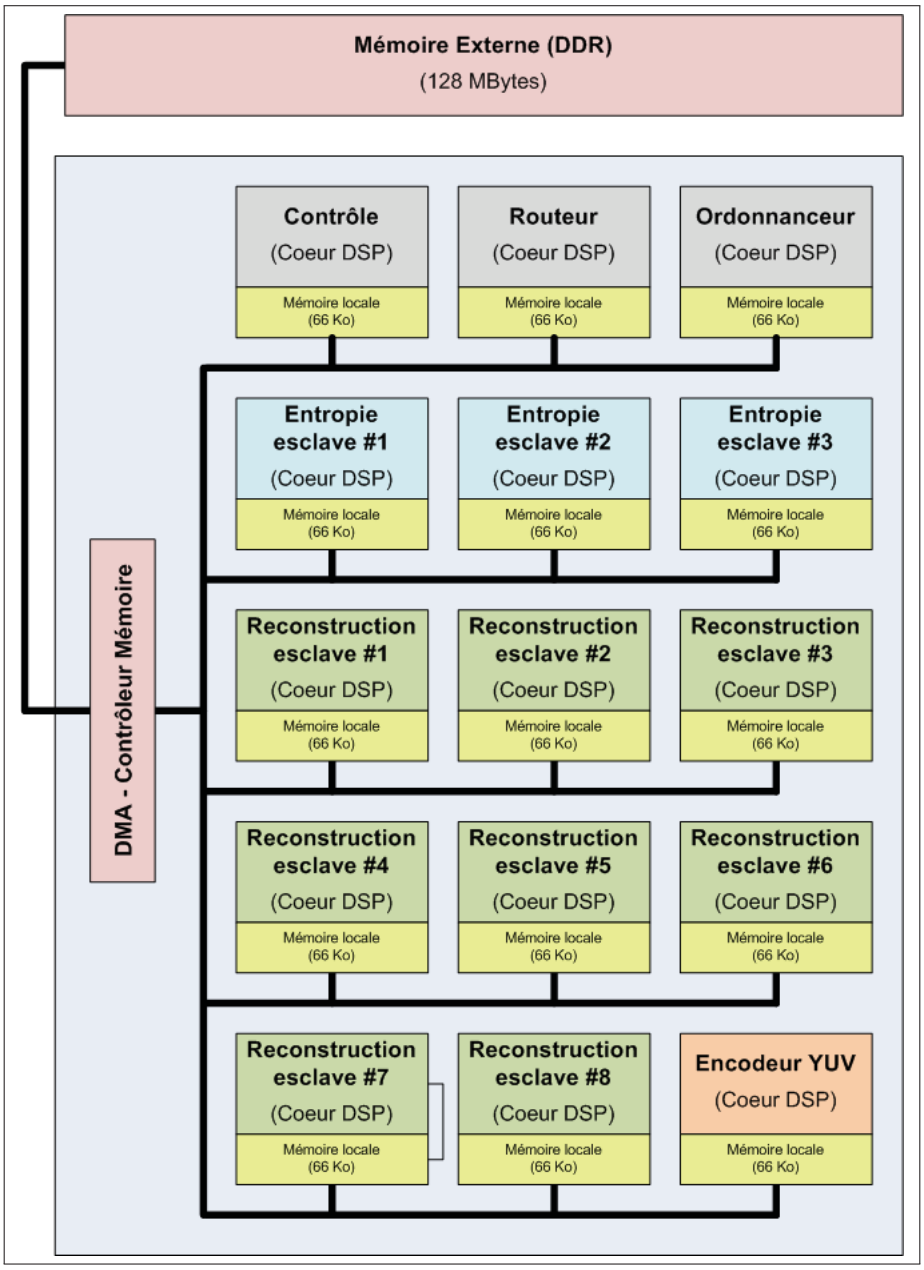


Figure 3.7 Schéma haut niveau du système sur le processeur DSP *OCT1010*.

3.2.1.2 Paramètres et membres du conteneur générique

Le conteneur générique encapsule les données du décodeur à l'aide d'une structure ayant neuf (9) principaux membres. Chacun de ces membres est énuméré et décrit dans la table 3.1.

Tableau 3.1 Description des principaux membres de la structure du conteneur générique

Type	Membres	Description
unsigned char *	pbyDDRContainerStartAddr	Adresse de départ de la zone de mémoire externe.
unsigned int	uiContainerSize	Taille de la structure de données pour un macrobloc.
void *	hTopLeftMbContainer	Pointeur sur les données locales du macrobloc en haut à gauche.
void *	hTopMbContainer	Pointeur sur les données locales du macrobloc en haut.
void *	hTopRightMbContainer	Pointeur sur les données locales du macrobloc en haut à droite.
void *	hLeftMbContainer	Pointeur sur les données locales du macrobloc à gauche.
void *	hCurrentMbContainer	Pointeur sur les données locales du macrobloc courant.
unsigned int	uiReadMask	Masque pour indiquer les positions des macroblocs étant requis en lecture.
unsigned int	uiWriteMask	Masque pour indiquer les positions des macroblocs étant requis en écriture.

Toutes les données d'un conteneur générique sont placées de manière contigüe en mémoire externe et tous les conteneurs génériques se retrouvent également de façon contigüe en mémoire externe. Cela fait en sorte qu'il est possible de représenter l'ensemble des conteneurs génériques pour une trame à l'aide d'une représentation spatiale. Le calcul pour obtenir le conteneur générique courant est donné par l'équation suivante :

$$CurMbContainer(X,Y) = [X + (Y * FrmWidth)] \times ContainerSize, \tag{3.1}$$

où $X \geq 0$ et $Y \geq 0$ représentent la position horizontale en macrobloc et la position verticale en macrobloc respectivement, *FrmWidth* représente la largeur de la trame en macrobloc, et *ContainerSize* représente la taille de la structure de données pour un macrobloc. La figure 3.8 illustre un exemple de la représentation spatiale en mémoire externe du mécanisme d'abstraction de la résolution. Dans cet exemple, la trame possède une largeur de 9 macroblocs et une hauteur de 6 macroblocs pour un total de 54 macroblocs. Le macrobloc jaune représente le macrobloc

courant à la position $(4, 2)$ et les flèches rouges illustrent les macroblocs voisins sur lesquels il y a des dépendances de données.

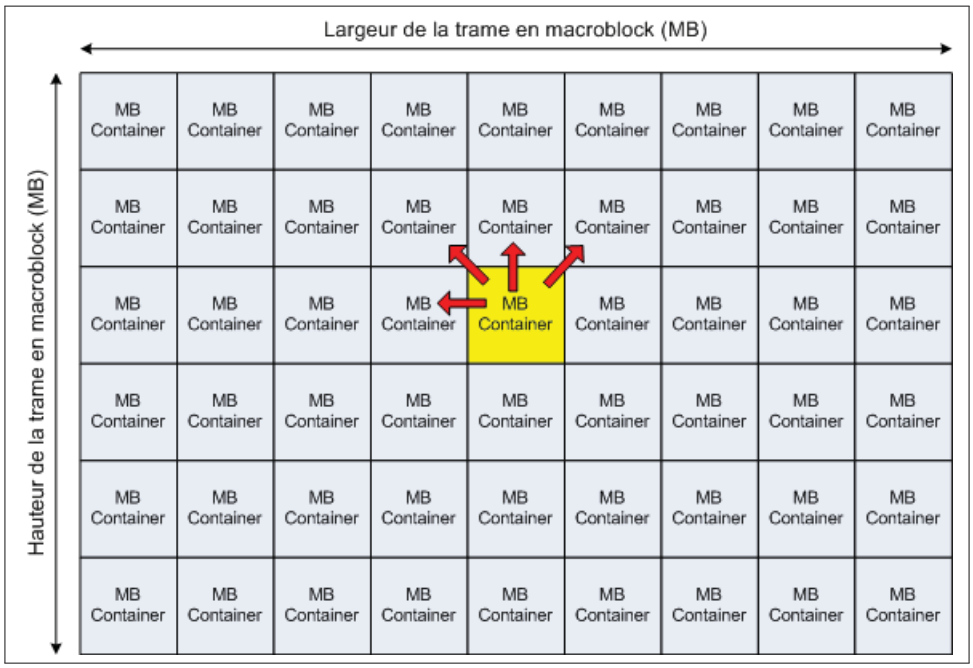


Figure 3.8 Exemple de la représentation spatiale en mémoire externe du mécanisme d'abstraction de la résolution.

3.2.1.3 Interface du conteneur générique

L'interface du conteneur générique offre trois appels de fonctions. La figure 3.9 illustre le diagramme du flot de contrôle pour l'interface du conteneur générique. Le premier appel de fonction est la remise à l'état initial. Cette fonction n'est appelée qu'une seule fois et sert à l'initialisation du mécanisme d'abstraction de la résolution. Lors de cette étape, tous les conteneurs génériques de macroblocs étant voisins au conteneur générique du macrobloc courant sont lus et transférés de la mémoire externe vers la mémoire locale. Le deuxième appel de fonction consiste à notifier le début de traitement d'un macrobloc. Lors de cette étape, les conteneurs génériques de macroblocs voisins n'étant pas présents en mémoire locale sont lus et transférés à partir de la mémoire externe. Finalement, le troisième et dernier appel de fonction consiste à notifier la fin du traitement d'un macrobloc. Lors de cette étape, les conteneurs génériques de macroblocs ayant été modifiés sont écrits en mémoire externe.

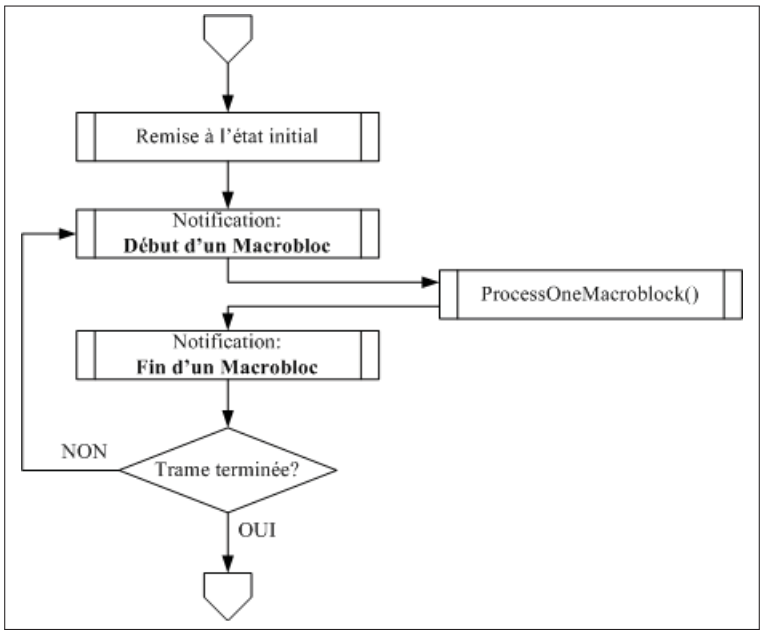


Figure 3.9 Diagramme du flot de contrôle de l'interface du conteneur générique.

3.2.1.4 Optimisation des transferts mémoires en lecture

L'utilisation des conteneurs génériques, servant à abstraire la résolution, permet d'optimiser les transferts mémoires en minimisant les lectures en mémoire externe. La figure 3.10 illustre un exemple de cette optimisation appliquée aux transferts mémoires. Lorsque le décodeur doit traiter le prochain macrobloc courant, c.-à-d. le macrobloc ayant l'indice $i+1$, l'ancien macrobloc courant, c.-à-d. le macrobloc ayant l'indice i, devient le macrobloc voisin de gauche du prochain macrobloc courant. Par la suite, le macrobloc voisin du haut ayant l'indice i devient le macrobloc voisin du haut à gauche de ce prochain macrobloc courant. Finalement, le macrobloc voisin du haut à droite ayant l'indice i devient le macrobloc voisin du haut de ce même prochain macrobloc courant. Cela fait en sorte que seul le macrobloc voisin du haut à droite ayant l'indice $i+1$ nécessite une lecture en mémoire externe. Si le prochain macrobloc courant se retrouve sur la bordure limite de droite, le macrobloc voisin du haut à droite ayant l'indice $i+1$ devient non-nécessaire. Bref, ce cas particulier en fin de ligne fait en sorte qu'aucune lecture en mémoire externe ne devient nécessaire pour traiter le prochain macrobloc courant qui se trouve en fait à être le dernier macrobloc de la ligne courante. Un début de ligne constitue aussi un cas particulier. Par contre, ce cas particulier nécessite beaucoup plus de lectures, car le

MB du haut ainsi que le MB courant de ce début de ligne ne sont pas disponibles en mémoire locale.

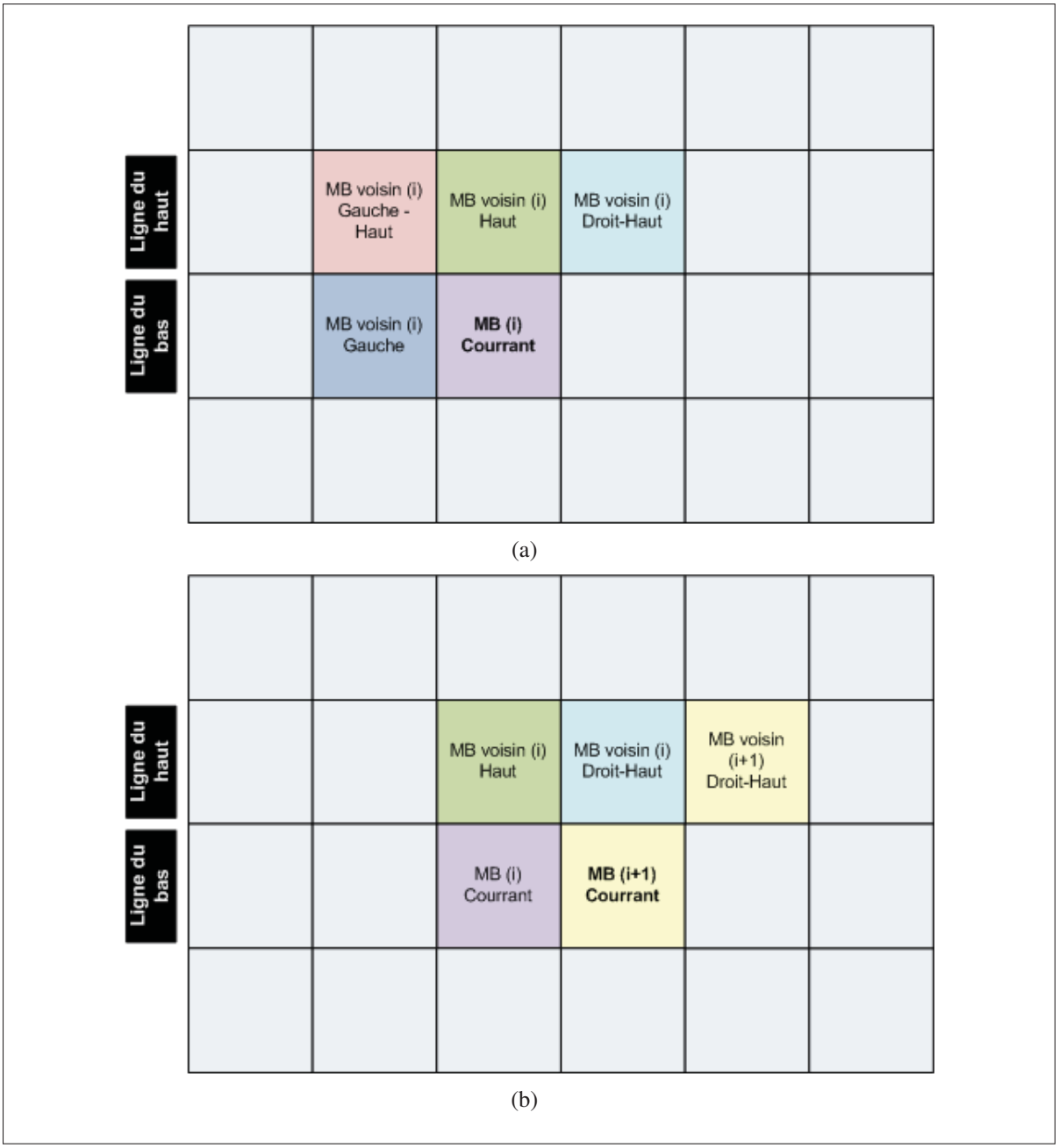


Figure 3.10 Illustration de l'optimisation des transferts mémoires en lecture (a) Mb indice i, (b) Mb indice i+1.

3.2.1.5 Conteneurs génériques utilisés par le décodeur

L'implémentation du décodeur H.264 utilise trois (3) conteneurs génériques. Le premier conteneur générique contient la structure utilisée pour représenter le contexte d'un macrobloc. Par la suite, le deuxième conteneur générique contient tous les coefficients provenant de la transformée H.264 pour un macrobloc. Finalement, le troisième conteneur générique contient tous les pixels reconstruits sur lesquels le filtrage antibloc fut appliqué pour un macrobloc. Le module du décodage de l'entropie et de l'analyse syntaxique ne requiert que l'utilisation des deux premiers conteneurs génériques, alors que le processus de la reconstruction et du filtrage antibloc requiert l'utilisation des trois conteneurs génériques. La figure 3.11 illustre le diagramme de l'utilisation des conteneurs génériques par les deux modules fonctionnels du décodeur.

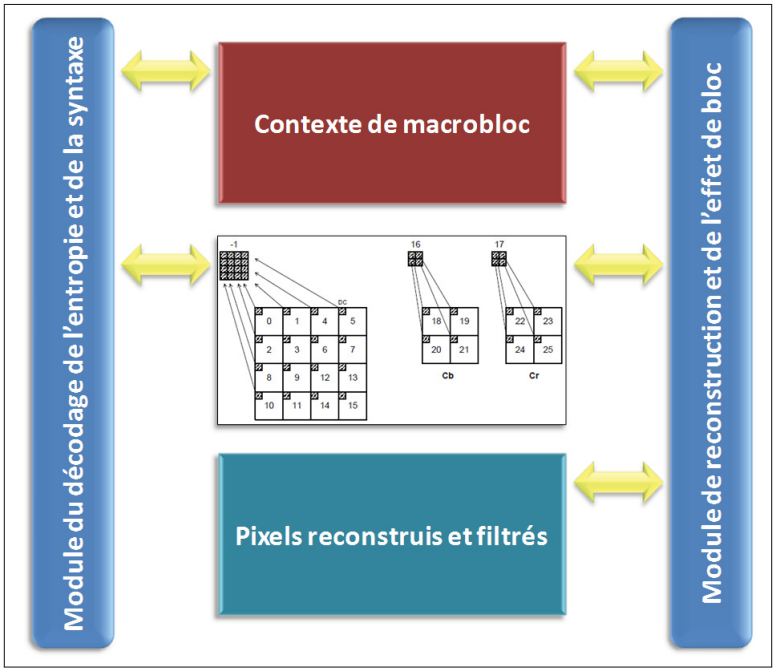


Figure 3.11 Diagramme des conteneurs génériques utilisés par les modules du décodeur.

3.2.2 Modèle de synchronisation et d'intercommunication

Comme illustré dans le schéma de la figure 3.2, l'architecture de l'implémentation du décodeur H.264 utilise un concept de *Maître/Esclave* (c.-à-d. *Master/Worker*) (Azevedo *et al.*, 2009b) (Azevedo *et al.*, 2009a) pour chacun des deux processus, soit le processus du décodage de

l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc. Les tâches effectuées par les *esclaves* faisant partie du processus du décodage de l'entropie et de l'analyse syntaxique (c.-à-d. décoder la syntaxe d'une ou plusieurs tranches) sont complètement indépendantes les unes des autres. Par contre, les tâches effectuées par les *esclaves* faisant partie du processus de la reconstruction et du filtrage antibloc (c.-à-d. reconstruire et filtrer des macroblocs) comportent des dépendances de données.

Pour demeurer générique, le modèle de synchronisation et d'intercommunication entre les *esclaves* doit tenir compte de ces différences entre les tâches. Les sous-sections 3.2.2.1 et 3.2.2.2 décrivent respectivement le comportement et les spécificités du modèle de synchronisation et d'intercommunication pour les tâches dites indépendantes et les tâches comportant des dépendances de données.

3.2.2.1 Tâches indépendantes

Le modèle de synchronisation pour les tâches dites indépendantes utilise une structure commune pour chacun des *esclaves*. Cette structure, se nommant *Worker Task Control* (WTC), se trouve en mémoire externe et est protégée par un mutex. Le module *maître* doit, à priori, initialiser le contenu de la structure WTC. Pour procéder à cette initialisation, le *maître* doit tout d'abord effectuer un balayage de tous les paquets appartenant à la trame vidéo actuelle pour y délimiter toutes les tranches s'y trouvant. Chaque tranche se retrouvant répertoriée dans la structure WTC constitue une tâche indépendante du décodage de l'entropie et de l'analyse syntaxique pour un *esclave*. Pour la deuxième et dernière étape de l'initialisation de la structure WTC, le *maître* doit démarrer (c.-à-d. réveiller) les *esclaves* par l'entremise de l'ordonnanceur du système. La formule du nombre d'*esclaves* à démarrer est donnée par :

$$num\_worker = \min(num\_slices, num\_worker\_available), \tag{3.2}$$

où num_slices représente le nombre de tranches présentes dans la trame vidéo courante, et $num\_worker\_available$ représente le nombre d'*esclaves* disponible au module *maître*. Lorsqu'un *esclave* est démarré, ce dernier doit accéder à la structure WTC pour obtenir une tâche

de décodage de l'entropie et de l'analyse syntaxique (c.-à-d. il doit obtenir une tranche à décoder). Lorsqu'un *esclave* a terminé sa tâche courante, il doit ré-accéder à la structure WTC pour obtenir sa prochaine tâche jusqu'à ce qu'aucune tâche ne soit plus disponible. Un effet d'auto-balancement des tâches pour le décodage de l'entropie entre les différents *esclaves* est présent en appliquant ce mécanisme lorsqu'il y a beaucoup plus de tranches que d'*esclaves*. Le diagramme de séquence de l'utilisation du WTC pour les tâches indépendantes les unes des autres par les *esclaves* est illustré à la figure 3.12.

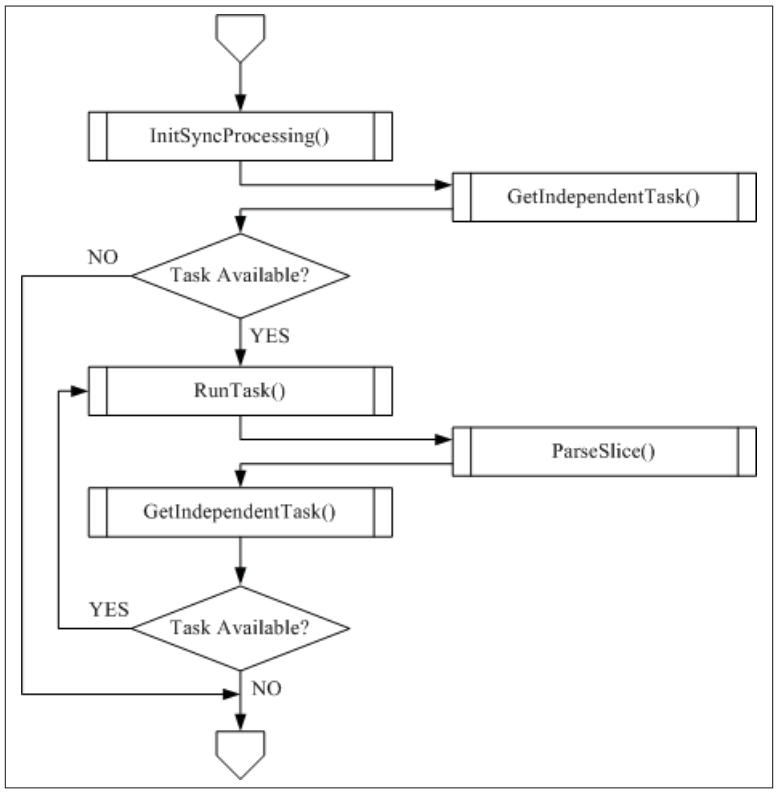


Figure 3.12 Diagramme de séquence de l'utilisation du WTC pour les tâches dites indépendantes les unes des autres.

3.2.2.2 Tâches avec dépendances de données

Le modèle de synchronisation pour les tâches avec dépendances de données utilise une structure commune pour chacun des *esclaves*, c.-à-d. le même modèle de structure utilisé par les tâches dites indépendantes, soit la structure se nommant WTC. Comme pour les tâches dites indépendantes, le module *maître* doit, à priori, initialiser le contenu de la structure WTC. Pour

procéder à cette initialisation, le *maître* doit tout d'abord remettre à zéro l'indice de la ligne de macroblocs à reconstruire et filtrer, indiquer les nombres de macroblocs présents sur une ligne, et indiquer l'intervalle en macroblocs servant à la notification de la dernière position du macrobloc reconstruit et filtré d'un *esclave(n)* vers un *esclave(n+1)*. Chaque ligne de macroblocs représente une tâche de reconstruction et de filtrage avec dépendances de données pour un *esclave*. Pour la deuxième et dernière étape de l'initialisation de la structure WTC, le *maître* doit démarrer les *esclaves*, c.-à-d. le nombre total d'*esclaves* disponibles au module *maître* selon la résolution courante de la trame vidéo, par l'entremise de l'ordonnanceur du système.

Lorsqu'un *esclave* est démarré, ce dernier doit accéder à la structure WTC pour obtenir une tâche de reconstruction et de filtrage (c.-à-d. il doit obtenir une ligne de macroblocs à reconstruire et à filtrer). Lors de l'exécution de la tâche courante, un *esclave(n)* doit attendre l'information de synchronisation provenant de l'*esclave(n-1)*, c.-à-d. l'*esclave* reconstruisant la ligne précédente, avant de reconstruire et filtrer chacun des macroblocs lui étant assignés. De plus, ce même *esclave(n)* doit fournir l'information de synchronisation à l'*esclave(n+1)*, c.-à-d. l'*esclave* reconstruisant la ligne suivante, après la reconstruction et le filtrage de chacun des macroblocs lui étant assignés. Lorsqu'un *esclave* a terminé de reconstruire une ligne de macroblocs (c.-à-d. lorsqu'il a terminé sa tâche courante), il doit réaccéder à la structure WTC pour obtenir sa prochaine tâche jusqu'à ce qu'aucune autre tâche ne soit disponible. Le diagramme de séquence de l'utilisation du WTC pour les tâches avec dépendances de données par les *esclaves* est illustré à la figure 3.13.

3.2.2.3 Protocole de communication entre les esclaves

Chaque *esclave* possède quatre boîtes de message (c.-à-d. *4 message box*) pour établir une communication entre l'*esclave* précédent et le prochain *esclave*. Il y a un premier canal de communication duplex entre l'*esclave(n)* et l'*esclave(n-1)*, c.-à-d. l'*esclave* précédent, qui se trouve à être le canal de communication en amont. Il y a un deuxième canal de communication duplex entre l'*esclave(n)* et l'*esclave(n+1)*, c.-à-d. le prochain *esclave*, qui se trouve à être le canal de communication en aval. Les deux boîtes de message utilisées pour le canal de communication en amont se nomment *PrevTxMsgBox* et *PrevRxMsgBox*, tandis que les deux

Figure 3.13 Diagramme de séquence de l'utilisation du WTC pour les tâches ayant des dépendances de données.

boîtes de message utilisées pour le canal de communication en aval se nomment *NextRxMsgBox* et *NextTxMsgBox*. Chacun des principaux membres de la structure d'une boîte de message pour les communications de synchronisation entre *esclaves* est décrit dans la table 3.2.

Le protocole de communication nécessite seulement un échange minimal de signaux et de notifications pour obtenir une synchronisation entre les différents *esclaves*. La figure 3.14 illustre le diagramme des échanges de signaux de communication entre *esclaves*.

Tableau 3.2 Description des principaux membres de la structure d'une boîte de message

Type	**Membres**	**Description**
void *	phyptWorkerMsgBox	Adresse physique de l'*esclave* ayant écrit dans cette boîte de message.
unsigned int	u1NtfyRegSignal	Signal envoyé par un *esclave(n+1)* pour s'enregistrer.
unsigned int	u14ProcMbIdx	Position du dernier macrobloc reconstruit et filtré par un *esclave(n-1)*.
unsigned int	u1ByeSignal	Signal envoyé par un *esclave(n+1)* pour informer son départ.
unsigned int	u1AckSignal	Signal envoyé par un *esclave(n-1)* pour informer la réception du signal u1ByeSignal.

Dans ce mécanisme d'intercommunication, un *esclave* courant doit obligatoirement s'enregistrer auprès de l'*esclave* précédent afin que ce dernier puisse lui rapporter, à intervalles réguliers (c.-à-d. selon la taille de l'intervalle de notification en macrobloc), la dernière position du macrobloc reconstruit et filtré. Pour effectuer cette tâche, l'*esclave* courant doit modifier les membres *phyptWorkerMsgBox* et *u1NtfyRegSignal* dans sa boîte de message *PrevTxMsgBox*. Ce message est ensuite envoyé dans la boîte de message *NextRxMsgBox* de l'*esclave* précédent. Ce dernier peut alors commencer à envoyer à l'*esclave* courant la position du dernier macrobloc reconstruit et filtré.

Lorsqu'un *esclave* courant reçoit une position de macrobloc se trouvant à être le dernier indice d'une ligne courante, ce dernier doit obligatoirement se déconnecter de l'*esclave* précédent. Pour effectuer cette tâche, l'*esclave* courant doit modifier le membre *u1ByeSignal* dans sa boîte de message *PrevTxMsgBox*. Lorsque l'*esclave* précédent reçoit le message de déconnexion via sa boîte *NextRxMsgBox*, ce dernier doit modifier le membre *u1AckSignal* dans sa boîte de message *NextTxMsgBox*. Ce message est envoyé dans la boîte de message *PrevRxMsgBox* de l'*esclave* courant. L'envoie de ce signal représente la dernière étape de la déconnexion de l'*esclave* courant avec son *esclave* précédent.

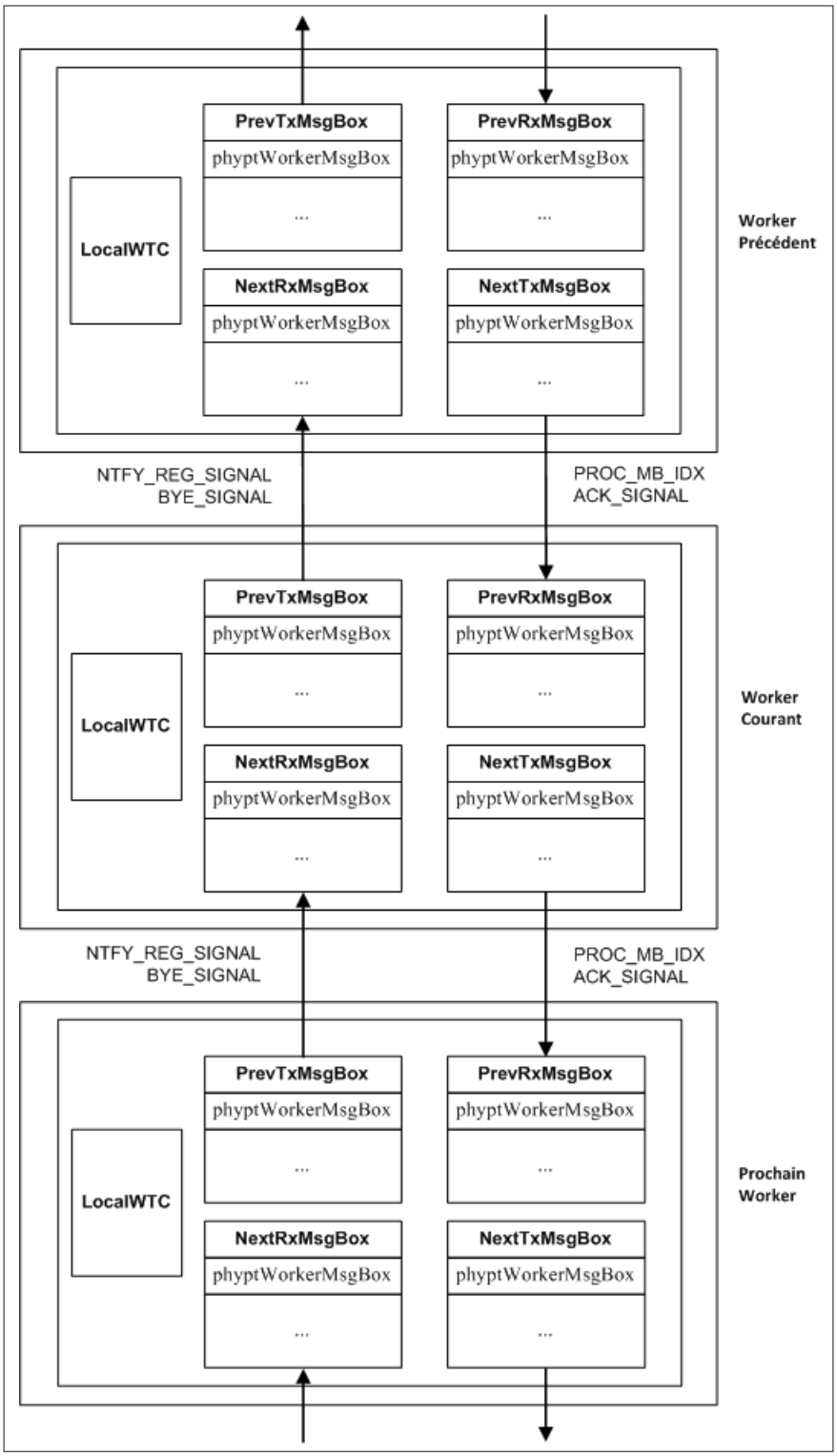


Figure 3.14 Diagramme des signaux de communication entre les esclaves.

3.3 Conclusion

Dans ce chapitre, la méthodologie appliquée pour l'implémentation du décodeur H.264 sur un processeur DSP multicoeurs fut présentée. Cette méthodologie utilise une approche hybride. En effet, cette méthodologie utilise une approche basée sur la fonctionnalité où chacune des fonctionnalités utilise une approche basée sur les données. Le décodeur utilise donc deux (2) fonctionnalités, soit le processus du décodage de l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc. De plus, pour répondre à la contrainte de temps réel, ces deux fonctionnalités sont pipelinées. L'utilisation du concept du partitionnement du décodeur H.264 en deux fonctionnalités fut très bénéfique au niveau de la taille du code. En effet, la mémoire locale sur un coeur DSP *OCT1010* est très limité et lorsque le décodeur est exécuté en un seul bloc de code, plusieurs *cache miss* se produisent lors de l'exécution, ce qui diminue grandement les performances du décodeur H.264. La taille du code de chacune des deux fonctionnalités est maintenant adéquate et optimale par rapport à la taille de la mémoire locale du coeur DSP *OCT1010*.

Pour pouvoir supporter diverses résolutions et pour pouvoir être distribuée sur plusieurs coeurs DSP, l'implémentation du décodeur H.264 supporte plusieurs concepts d'extensibilité. La méthodologie présentée dans ce chapitre utilise un mécanisme d'abstraction de la résolution pour rendre le modèle mémoire extensible. De plus, la méthodologie employée pour l'implémentation du décodeur utilise un modèle de synchronisation et d'intercommunication générique étant applicable pour des tâches indépendantes et des tâches avec dépendances de données tout en étant extensible par rapport au nombre de coeurs DSP disponibles. La prochaine étape, présentée au chapitre 3, vise à optimiser plus spécifiquement une des étapes du décodage de l'entropie de type CAVLC, soit une zone critique du module du décodage de l'entropie et de l'analyse syntaxique qui est aussi la première fonctionnalité présentée dans ce chapitre.

CHAPITRE 4

OPTIMISATION DU DÉCODAGE DE L'ENTROPIE DE TYPE CAVLC

Selon la spécification H.264 (ITU, 2007), l'algorithme de décodage de l'entropie de type CAVLC est composé de cinq (5) étapes. En effet, comme décrit par Moon *et al.* (2005), les coefficients quantifiés et réordonnancés avec un patron zigzagué sont compressés à l'aide de cinq éléments de syntaxe : *coeff_token*, signe des *T1s*, *level*, *total_zeros*, et *run_before*. La figure 4.1 illustre les cinq étapes de cet algorithme pour extraire les cinq éléments permettant de retrouver la matrice de coefficients quantifiés et zigzagués. Le décodage de l'entropie de type CAVLC représente la fonctionnalité ayant le plus grand temps total d'exécution pour le module du décodage de l'entropie et de la syntaxe. Comme ce processus se fait de manière séquentielle et que la seule façon de paralléliser ce dernier provient du concept de tranches à l'intérieur d'une trame vidéo, il devient donc primordial que le décodage de l'entropie de type CAVLC soit le plus efficace possible.

Plusieurs méthodes pour optimiser le décodage de l'entropie de type CAVLC, basées sur les travaux de Iqbal et Henkel (2009) et de Kim *et al.* (2006), sont déjà présentes à l'intérieur de l'implémentation du décodeur H.264 utilisé par ce travail. Par contre, aucune de ces méthodes n'améliore le temps de décodage des mots de code à longueur variable pour l'élément *total_zeros* qui est la quatrième étape du décodage de l'entropie de type CAVLC. La méthode de décodage qui est implémentée par le code de référence du décodeur H.264 pour l'extraction du code *total_zeros* provient d'un algorithme de recherche séquentielle dans une table de correspondance qui est peu performante au niveau de la vitesse d'exécution. Un nouvel algorithme tirant avantage du décodage arithmétique pour l'extraction du nombre total de zéros à l'intérieur d'un bloc 4x4, soit l'élément *total_zeros*, est proposé dans ce chapitre. Les résultats de l'accélération du décodage de type CAVLC à l'aide de ce nouvel algorithme seront présentés au chapitre 5 (section 5.2).

Ce chapitre est divisé en quatre sections principales. À la section 4.1, nous décrivons les cinq (5) étapes du décodage de l'entropie de type CAVLC, alors qu'à la section 4.2, nous abordons

Figure 4.1 Diagramme de l'entropie de type CAVLC.

plus particulièrement l'étape 4 du décodage de type CAVLC, soit l'extraction du nombre total de zéros dans un bloc 4x4 qui est aussi appelé élément *total_zeros*. Par la suite, à la section 4.3, nous décrivons le nouvel algorithme pour l'extraction du nombre total de zéros dans un bloc 4x4. Pour terminer, nous concluons sur les points discutés dans ce chapitre à la section 4.4.

4.1 Les cinq étapes du décodage de l'entropie de type CAVLC

Tout d'abord, la première étape du décodage de l'entropie de type CAVLC consiste à extraire le code se nommant *coeff_token* (Iqbal et Henkel, 2009; Kim *et al.*, 2006; Moon, 2007). Ce code se réfère à la table, provenant de la spécification H.264. Ce code permet d'obtenir le nombre total de coefficients non nuls, ainsi que le nombre de coefficients ayant une valeur de 1 ou -1, aussi appelés *trailing_ones*, à la fin du vecteur de coefficients zigzagués.

Ensuite, la deuxième étape consiste à extraire le signe de chacun des coefficients ayant la valeur 1 ou -1 (*trailing_ones*). Seulement un bit est utilisé pour représenter le signe de chacun de ces coefficients, où la valeur 1 représente un nombre négatif, alors que la valeur 0 représente un

nombre positif. Si le nombre total de coefficients non nuls est de zéro, ou bien si le nombre de *trailing_ones* est de zéro, cette étape n'est pas effectuée.

Par la suite, la troisième étape consiste à extraire le niveau fréquentiel des coefficients non nuls. Chaque niveau fréquentiel est composé d'un préfixe ainsi que d'un suffixe, aussi appelé *level_prefix* et *level_suffix*. Cette étape est facultative. Si le nombre de coefficients non nuls, extrait du code *coeff_token*, est de zéro, cette étape n'est pas effectuée.

Puis, la quatrième étape consiste à extraire le nombre total de zéros situés avant le dernier coefficient non nul dans le bloc courant (Kim *et al.*, 2006; Moon *et al.*, 2008; Moon, 2008). Ce code se nomme *total_zeros*. Cette étape est effectuée seulement si le nombre total de coefficients non nuls, extrait du code *coeff_token*, est plus petit que le nombre total de coefficients que peut contenir le bloc courant.

Finalement, la cinquième et dernière étape du décodage de l'entropie de type CAVLC consiste à extraire les suites consécutives du nombre zéro précédent un coefficient non nul (Kim *et al.*, 2006; Lee *et al.*, 2008). Ce code est aussi nommé *run_before*. Cette étape n'est pas effectuée si le nombre total de zéros ainsi que le nombre total de coefficients non nuls sont de zéro.

4.2 Extraction du nombre total de zéros pour un bloc 4x4

Comme il est mentionné auparavant dans ce chapitre, l'extraction du nombre total de zéros (*total_zeros*) pour un bloc 4x4 est la quatrième étape du décodage de l'entropie de type CAVLC. Cet élément représente le nombre de coefficients nuls précédant le coefficient non nul ayant la plus haute fréquence à l'intérieur de la matrice réordonnancée. Les tableaux 4.1 et 4.2 décrivent la syntaxe des 15 tables servant à extraire l'élément *total_zeros* en fonction du nombre total de coefficients non nuls (*TotalCoeff*), où la variable *TotalCoeff* provient de l'extraction de l'élément *coeff_token* obtenu lors de la première étape du décodage.

La méthode de décodage implémentée par le code de référence du décodeur H.264 pour l'extraction du code *total_zeros* provient d'un algorithme de recherche séquentielle dans une table de correspondance. Cette méthode est aussi connue sous le nom de *Table Look-up by Sequen-*

Tableau 4.1 Éléments *total_zeros* pour les blocs 4x4 avec TotalCoeff(*coeff_token*) de 1 à 7.
Tiré de ITU (2007).

total_zeros	TotalCoeff(*coeff_token*)						
	1	2	3	4	5	6	7
0	1	111	0101	0001 1	0101	0000 01	0000 01
1	011	110	111	111	0100	0000 1	0000 1
2	010	101	110	0101	0011	111	101
3	0011	100	101	0100	111	110	100
4	0010	011	0100	110	110	101	011
5	0001 1	0101	0011	101	101	100	11
6	0001 0	0100	100	100	100	011	010
7	0000 11	0011	011	0011	011	010	0001
8	0000 10	0010	0010	011	0010	0001	001
9	0000 011	0001 1	0001 1	0010	0000 1	001	0000 00
10	0000 010	0001 0	0001 0	0001 0	0001	0000 00	
11	0000 0011	0000 11	0000 01	0000 1	0000 0		
12	0000 0010	0000 10	0000 1	0000 0			
13	0000 0001 1	0000 01	0000 00				
14	0000 0001 0	0000 00					
15	0000 0000 1						

Tableau 4.2 Éléments *total_zeros* pour les blocs 4x4 avec TotalCoeff(*coeff_token*) de 8 à 15.
Tiré de ITU (2007).

total_zeros	TotalCoeff(*coeff_token*)							
	8	9	10	11	12	13	14	15
0	0000 01	0000 01	0000 1	0000	0000	000	00	0
1	0001	0000 00	0000 0	0001	0001	001	01	1
2	0000 1	0001	001	001	01	1	1	
3	011	11	11	010	1	01		
4	11	10	10	1	001			
5	10	001	01	011				
6	010	01	0001					
7	001	0000 1						
8	0000 00							

cial Search (TLSS). La méthode TLSS retourne le code *total_zeros* en extrayant qu'un seul bit à la fois du train de bits de la trame vidéo. À chaque extraction, un accès mémoire est exécuté pour vérifier si le mot de code correspond au contenu de la table de correspondance. Si le mot de code ne correspond pas au contenu de cette table de correspondance, une autre itération est alors effectuée en extrayant un autre bit du train de bits de la trame vidéo, jusqu'à ce qu'il y ait

une correspondance entre le mot de code et le contenu de la table de correspondance. Chaque accès mémoire dégrade les performances de l'algorithme, car un accès mémoire possède un temps d'exécution plus lent que le temps d'exécution d'une instruction par une unité arithmétique et logique (ALU) faisant partie d'un coeur DSP. De plus, l'exécution de cet algorithme est considérée comme étant très lente (Kim *et al.*, 2006; Iqbal et Henkel, 2009).

4.3 Nouvel algorithme pour l'extraction du nombre total de zéros dans un bloc 4x4

Cette section a pour objectif de trouver un nouvel algorithme pour l'extraction du nombre total de zéros dans un bloc 4x4. La sous-section 4.3.1 présente des méthodes avancées pour l'extraction du code *total_zeros*, alors que la sous-section 4.3.2 présente une nouvelle méthode pour l'extraction de ce même code.

4.3.1 Méthodes avancées pour l'extraction du code *total_zeros*

Un premier algorithme est proposé par Kim *et al.* (2006) pour extraire plus adéquatement l'élément de syntaxe *total_zeros*. L'extrait 4.1 illustre le pseudo-code de cet algorithme qui applique une technique se nommant décodage arithmétique. Par contre, le décodage arithmétique est appliqué seulement sur les groupes provenant des tables de *TotalCoeff* 1, 13, 14 et 15, car aucun lien arithmétique ne fut trouvé pour les autres tables. Cette méthode propose donc d'utiliser l'algorithme fourni dans le code de référence sur les groupes provenant des tables de *TotalCoeff* allant de 2 à 12 inclusivement.

Un deuxième algorithme est proposé par Moon (2008) pour extraire plus adéquatement l'élément de syntaxe *total_zeros*. Cette méthode propose de remplacer les multiples accès en mémoire par de simples opérations mathématiques et de réorganiser le contenu mémoire des tables de correspondance lorsque les opérations arithmétiques ne sont pas autorisées. Pour élaborer leur solution, les auteurs de l'article Moon (2008) ont réarrangé les mots de code par rapport au nombre de bits 0 suivis par le premier bit 1 dans le mot de code provenant de la spécification H.264. La table 4.3 illustre le réarrangement des codes VLC pour l'extraction des éléments *total_zeros*, où TZ représente *total_zeros* et CW représente *code word*. Ce réarrangement de

Extrait 4.1 Pseudo algorithme pour le décodage de l'élément *total_zeros*.
Tiré de Kim *et al.* (2006).

```
if(TC == 1) {
    code = ShowBits(9);

    if(code >= 256) { TZ = 0; n_skip = 1; }
    else if(code >= 4) {
        m = GetM(code, 9);
        TZ = (m << 1) - ((code >> (7 - m)) & 1);
        n_skip = m + 2; }
    else { TZ = 16 - code; n_skip = 9; }
}
else if(TC <= 12) {
    /* use reference code function */
}
else if(TC == 13) {
    code = ShowBits(3); g = code >> 2;
    h = code >> (g + 1); TZ = ((code>>g) & (h + 1)) + (h - g);
    n_skip = 3 - g - h;
}
else {  /* TC == 14 or 15 */
    code = ShowBits(16 - TC); g = code >> 1;
    TZ = code & (g + 1); n_skip = 16 - TC - g;
}
```

mots de code a largement contribué à inspirer la nouvelle méthode proposée dans le présent travail.

4.3.2 Nouvelle méthode proposée pour l'extraction du code *total_zeros*

Une nouvelle méthode est proposée par le présent travail pour extraire les mots de code *total_zeros* d'une trame vidéo compressée à l'aide de la spécification H.264. Cette nouvelle méthode réutilise, entre autres, l'algorithme présenté par Kim *et al.* (2006) pour le décodage arithmétique appliqué sur les groupes provenant des tables de *TotalCoeff* 1, 13, 14 et 15. Cette méthode applique un nouveau décodage arithmétique sur les groupes provenant des tables de *TotalCoeff* allant de 2 à 12 inclusivement. Pour parvenir à cette nouvelle méthode, cet algorithme s'est largement inspiré de la réorganisation de la table de correspondance présentée par le travail de Moon (2008) (voir la table 4.3).

La sous-section 4.3.2.1 présente un nouvel arrangement des tables VLC pour extraire le mot de code *total_zeros*. Suite à l'analyse de ce réarrangement, les sous-sections 4.3.2.2 et 4.3.2.3 présentent trois (3) nouvelles tables de correspondance pour extraire le mot de code *total_zeros*.

Tableau 4.3 Codes VLC basés sur le réarrangement des éléments *total_zeros* (TZ), où Tc représente l'élément TotalCoeff(*coeff_token*) et CW représente *code word*. Tiré de Moon (2008).

Tc = 1		Tc = 2		Tc = 3		Tc = 4		Tc = 5	
TZ	CW	TZ	CW	TZ	CW	TZ	CW	TZ	CW
0	1	0	111	1	111	1	111	3	111
1	011	1	110	2	110	4	110	4	110
2	010	2	101	3	101	5	101	5	101
3	0011	3	100	6	100	6	100	6	100
4	0010	4	011	7	011	8	011	7	011
5	00011	5	0101	0	0101	2	0101	0	0101
6	00010	6	0100	4	0100	3	0100	1	0100
7	000011	7	0011	5	0011	7	0011	2	0011
8	000010	8	0010	8	0010	9	0010	8	0010
9	0000011	9	00011	9	00011	0	00011	10	0001
10	0000010	10	00010	10	00010	10	00010	9	00001
11	00000011	11	000011	12	00001	11	00001	11	00000
12	00000010	12	000010	11	000001	12	00000		
13	000000011	13	000001	13	000000				
14	000000010	14	000000						
15	000000001								

Tc = 6		Tc = 7		Tc = 8		Tc = 9		Tc = 10	
TZ	CW	TZ	CW	TZ	CW	TZ	CW	TZ	CW
2	111	5	11	4	11	3	11	3	11
3	010	2	101	5	10	4	10	4	10
4	101	3	100	3	011	6	01	5	01
5	100	4	011	6	010	5	001	2	001
6	011	6	010	7	001	2	0001	6	0001
7	010	8	001	1	0001	7	00001	0	00001
9	001	7	0001	2	00001	0	000001	1	00000
8	0001	1	00001	0	000001	1	000000		
1	00001	0	000001	8	000000				
0	000001	9	000000						
10	000000								

Tc = 11		Tc = 12		Tc = 13		Tc = 14		Tc = 15	
TZ	CW	TZ	CW	TZ	CW	TZ	CW	TZ	CW
4	1	3	1	2	1	2	1	1	1
5	011	2	01	3	01	1	01	0	0
3	010	4	001	1	001	0	00		
2	001	1	0001	0	000				
1	0001	0	0000						
0	0000								

Pour terminer, la sous-section 4.3.2.4 présente le pseudo-code de la nouvelle méthode proposée qui réduit le niveau de complexité de l'algorithme pour un processeur DSP.

4.3.2.1 Analyse de la table VLC pour l'extraction du code *total_zeros*

Les tables 4.4, 4.5, 4.6, 4.7, 4.8, et 4.9 utilisent exactement le même réarrangement proposé par l'article de Moon (2008). Par contre, quatre (4) nouvelles colonnes ont été rajoutées pour trouver une nouvelle solution pour l'extraction des mots de code *total_zeros* qui favoriserait

l'utilisation d'opérations arithmétiques. La première nouvelle colonne contient le mot de code représenté sous sa forme décimale (ou bien *Decimal Code Word*). La deuxième nouvelle colonne contient le mot de code étendu (ou bien *Extended Code Word*) avec des bits 0 ajoutés à la fin pour que tous les membres de cette colonne aient le même nombre de bits. La troisième nouvelle colonne contient le mot de code étendu représenté sous sa forme décimale (ou bien *Decimal Extended Code Word*). La quatrième et dernière nouvelle colonne contient le groupe (Grp) en termes du nombre de bits 0 rajoutés au CW pour former la colonne ECW.

Tableau 4.4 Éléments *total_zeros* avec TotalCoeff(*coeff_token*) de 2 et 3.

TotalCoeff(*coeff_token*) = 2						TotalCoeff(*coeff_token*) = 3					
TZ	CW	DCW	ECW	DECW	Grp	TZ	CW	DCW	ECW	DECW	Grp
0	111	7	**111**000	56	3	1	111	7	**111**000	56	3
1	110	6	**110**000	48	3	2	110	6	**110**000	48	3
2	101	5	**101**000	40	3	3	101	5	**101**000	40	3
3	100	4	**100**000	32	3	6	100	4	**100**000	32	3
4	011	3	**011**000	24	3	7	011	3	**011**000	24	3
5	0101	5	**0101**00	20	2	0	0101	5	**0101**00	20	2
6	0100	4	**0100**00	16	2	4	0100	4	**0100**00	16	2
7	0011	3	**0011**00	12	2	5	0011	3	**0011**00	12	2
8	0010	2	**0010**00	8	2	8	0010	2	**0010**00	8	2
9	00011	3	**00011**0	6	1	9	00011	3	**00011**0	6	1
10	00010	2	**00010**0	4	1	10	00010	2	**00010**0	4	1
11	000011	3	**000011**	3	0	12	00001	1	**00001**0	2	1
12	000010	2	**000010**	2	0	11	000001	1	**000001**	1	0
13	000001	1	**000001**	1	0	13	000000	0	**000000**	0	0
14	000000	0	**000000**	0	0						

Tableau 4.5 Éléments *total_zeros* avec TotalCoeff(*coeff_token*) de 4 et 5.

TotalCoeff(*coeff_token*) = 4						TotalCoeff(*coeff_token*) = 5					
TZ	CW	DCW	ECW	DECW	Grp	TZ	CW	DCW	ECW	DECW	Grp
1	111	7	**111**00	28	2	3	111	7	**111**00	28	2
4	110	6	**110**00	24	2	4	110	6	**110**00	24	2
5	101	5	**101**00	20	2	5	101	5	**101**00	20	2
6	100	4	**100**00	16	2	6	100	4	**100**00	16	2
8	011	3	**011**00	12	2	7	011	3	**011**00	12	2
2	0101	5	**0101**0	10	1	0	0101	5	**0101**0	10	1
3	0100	4	**0100**0	8	1	1	0100	4	**0100**0	8	1
7	0011	3	**0011**0	6	1	2	0011	3	**0011**0	6	1
9	0010	2	**0010**0	4	1	8	0010	2	**0010**0	4	1
0	00011	3	**00011**	3	0	10	0001	1	**0001**0	2	1
10	00010	2	**00010**	2	0	9	00001	1	**00001**	1	0
11	00001	1	**00001**	1	0	11	00000	0	**00000**	0	0
12	00000	0	**00001**	0	0						

Tableau 4.6 Éléments *total_zeros* avec TotalCoeff(*coeff_token*) de 6.

TotalCoeff(*coeff_token*) = 6					
TZ	CW	DCW	ECW	DECW	Grp
2	111	7	**111**000	56	3
3	110	6	**110**000	48	3
4	101	5	**101**000	40	3
5	100	4	**100**000	32	3
6	011	3	**011**000	24	3
7	010	2	**010**000	16	3
9	001	1	**001**000	8	3
8	0001	1	**0001**00	4	2
1	00001	1	**00001**0	2	1
0	000001	1	**000001**	1	0
10	000000	0	**000000**	0	0

Tableau 4.7 Éléments *total_zeros* avec TotalCoeff(*coeff_token*) de 7 et 8.

TotalCoeff(*coeff_token*) = 7						TotalCoeff(*coeff_token*) = 8					
TZ	CW	DCW	ECW	DECW	Grp	TZ	CW	DCW	ECW	DECW	Grp
5	11	3	**11**0000	48	4	4	11	3	**11**0000	48	4
2	101	5	**101**000	40	3	5	10	2	**10**0000	32	4
3	100	4	**100**000	32	3	3	011	3	**011**000	24	3
4	011	3	**011**000	24	3	6	010	2	**010**000	16	3
6	010	2	**010**000	16	3	7	001	1	**001**000	8	3
8	001	1	**001**000	8	3	1	0001	1	**0001**00	4	2
7	0001	1	**0001**00	4	2	2	00001	1	**00001**0	2	1
1	00001	1	**00001**0	2	1	0	000001	1	**000001**	1	0
0	000001	1	**000001**	1	0	8	000000	0	**000000**	0	0
9	000000	0	**000000**	0	0						

Il est possible d'observer au travers des colonnes DCW des divers tableaux présentés dans cette sous-section que les valeurs oscillent entre 0 et 7 inclusivement. Plus spécifiquement, les valeurs DCW des tableaux 4.4, 4.5, et 4.6 oscillent entre 0 et 7 inclusivement. Tandis que les valeurs DCW des tableaux 4.7, 4.8, et 4.9 oscillent entre 0 et 5 inclusivement. Cette relation est utilisée pour générer les deux (2) tables de correspondances (LUT) présentées à la sous-section 4.3.2.2.

Par la suite, il est possible de trouver une corrélation entre les colonnes DECW et Grp des divers tableaux présentés dans cette sous-section pour les différents groupes provenant de TotalCoeff(*coeff_token*). Les valeurs de Grp correspondent au nombre de bits 0 ajoutés à la fin

Tableau 4.8 Éléments *total_zeros* avec TotalCoeff(*coeff_token*) de 9 et 10.

TotalCoeff(*coeff_token*) = 9						TotalCoeff(*coeff_token*) = 10					
TZ	CW	DCW	ECW	DECW	Grp	TZ	CW	DCW	ECW	DECW	Grp
3	11	3	**11**0000	48	4	3	11	3	**11**000	24	3
4	10	2	**10**0000	32	4	4	10	2	**10**000	16	3
6	01	1	**01**0000	16	4	5	01	1	**01**000	8	3
5	001	1	**001**000	8	3	2	001	1	**001**00	4	2
2	0001	1	**0001**00	4	2	6	0001	1	**0001**0	2	1
7	00001	1	**00001**0	2	1	0	00001	1	**00001**	1	0
0	000001	1	**000001**	1	0	1	00000	0	**00000**	0	0
1	000000	0	**000000**	0	0						

Tableau 4.9 Éléments *total_zeros* avec TotalCoeff(*coeff_token*) de 11 et 12.

TotalCoeff(*coeff_token*) = 11						TotalCoeff(*coeff_token*) = 12					
TZ	CW	DCW	ECW	DECW	Grp	TZ	CW	DCW	ECW	DECW	Grp
4	1	1	**1**000	8	3	3	1	1	**1**000	8	3
5	011	3	**011**0	6	1	2	01	1	**01**00	4	2
3	010	2	**010**0	4	1	4	001	1	**001**0	2	1
2	001	1	**001**0	2	1	1	0001	1	**0001**	1	0
1	0001	1	**0001**	1	0	0	0000	0	**0000**	0	0
0	0000	0	**0000**	0	0						

des membres de la colonne CW pour que tous les membres de la colonne ECW aient le même nombre de bits. Les caractères en gras pour les membres de la colonne ECW représentent donc les mots de code de la colonne CW. Les colonnes DECW servent à déterminer les bornes supérieures et inférieures des groupes (Grp) auxquels appartiennent les mots de code (CW). Cette corrélation est utilisée pour générer la table de contrôle dans la sous-section 4.3.2.3 ainsi que pour définir la variable *right_shift* présentée à l'équation 4.8.

Par exemple, lorsque $TC = 10$, les groupes (Grp) 0, 1, 2 et 3 possèdent les bornes $[0,1]$, $[2,3]$, $[4,7]$ et $[8,31]$ respectivement. La borne inférieure d'un groupe provient donc de la plus petite valeur DECW correspondante à la colonne Grp, tandis que la borne supérieure d'un groupe provient de la plus grande valeur DECW correspondante à la colonne Grp du même groupe.

4.3.2.2 Génération de deux tables de correspondance

Cette sous-section présente deux (2) nouvelles tables de correspondance pour obtenir les valeurs *total_zeros*. Ces deux nouvelles tables de correspondance sont élaborées à partir des observations faites à l'intérieur de la sous-section 4.3.2.1, soit les observations faites sur les colonnes DCW des tableaux présentés dans cette sous-section. Le tableau 4.10 définit la première table de correspondance (LUT) pour obtenir les valeurs *total_zeros* (TZ) lorsque le paramètre *TotalCoeff*(*coeff_token*) (TC) se trouve dans l'intervalle $[2,6]$. Chaque élément du tableau occupe une taille de 4 bits et la taille résultante de cette table de correspondance est de 80 octets (5 sous-tableaux de 8 colonnes par 4 lignes contenant un élément d'un demi-octet).

Tableau 4.10 LUT pour obtenir les valeurs TZ avec TotalCoeff(*coeff_token*) de 2 à 6

TotalCoeff(*coeff_token*) = 2								
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)	TZ (CW=6)	TZ (CW=7)
0	14	13	12	11	-	-	-	-
1	-	-	10	9	-	-	-	-
2	-	-	8	7	6	5	-	-
3	-	-	-	4	3	2	1	0
TotalCoeff(*coeff_token*) = 3								
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)	TZ (CW=6)	TZ (CW=7)
0	13	11	-	-	-	-	-	-
1	-	12	10	9	-	-	-	-
2	-	-	8	5	4	0	-	-
3	-	-	-	7	6	3	2	1
TotalCoeff(*coeff_token*) = 4								
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)	TZ (CW=6)	TZ (CW=7)
0	12	11	10	0	-	-	-	-
1	-	-	9	7	3	2	-	-
2	-	-	-	8	6	5	4	1
3	-	-	-	-	-	-	-	-
TotalCoeff(*coeff_token*) = 5								
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)	TZ (CW=6)	TZ (CW=7)
0	11	9	-	-	-	-	-	-
1	-	10	8	2	1	0	-	-
2	-	-	-	7	6	5	4	3
3	-	-	-	-	-	-	-	-
TotalCoeff(*coeff_token*) = 6								
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)	TZ (CW=6)	TZ (CW=7)
0	10	0	-	-	-	-	-	-
1	-	1	-	-	-	-	-	-
2	-	8	-	-	-	-	-	-
3	-	9	7	6	5	4	3	2

Par la suite, le tableau 4.11 définit la deuxième table de correspondance (LUT) pour obtenir les valeurs *total_zeros* (TZ) lorsque *TotalCoeff*(*coeff_token*) (TC) se trouve dans l'intervalle

[7, 12]. Chaque élément du tableau occupe une taille de 4 bits et la taille résultante de cette table de correspondance est de 90 octets (6 sous-tableaux de 6 colonnes par 5 lignes contenant un élément d'un demi-octet).

Tableau 4.11 LUT pour obtenir les valeurs TZ avec TotalCoeff(*coeff_token*) de 7 à 12

TotalCoeff(*coeff_token*) = 7						
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)
0	9	0	-	-	-	-
1	-	1	-	-	-	-
2	-	7	-	-	-	-
3	-	8	6	4	3	2
4	-	-	-	5	-	-
TotalCoeff(*coeff_token*) = 8						
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)
0	8	0	-	-	-	-
1	-	2	-	-	-	-
2	-	1	-	-	-	-
3	-	7	6	3	-	-
4	-	-	5	4	-	-
TotalCoeff(*coeff_token*) = 9						
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)
0	1	0	-	-	-	-
1	-	7	-	-	-	-
2	-	2	-	-	-	-
3	-	5	-	-	-	-
4	-	6	4	3	-	-
TotalCoeff(*coeff_token*) = 10						
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)
0	1	0	-	-	-	-
1	-	6	-	-	-	-
2	-	2	-	-	-	-
3	-	5	4	3	-	-
4	-	-	-	-	-	-
TotalCoeff(*coeff_token*) = 11						
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)
0	0	1	-	-	-	-
1	-	2	3	5	-	-
2	-	-	-	-	-	-
3	-	4	-	-	-	-
4	-	-	-	-	-	-
TotalCoeff(*coeff_token*) = 12						
Offset	TZ (CW=0)	TZ (CW=1)	TZ (CW=2)	TZ (CW=3)	TZ (CW=4)	TZ (CW=5)
0	0	1	-	-	-	-
1	-	4	-	-	-	-
2	-	2	-	-	-	-
3	-	3	-	-	-	-
4	-	-	-	-	-	-

Les équations pour accéder à un élément de ces deux tables de correspondance sont fournies par :

$$base\_address = [(((TC - start\_TC) \times 1stLevelOffset) + CW) \times 2ndLevelOffset], \quad (4.1)$$

$$offset = [right\_shift], \quad (4.2)$$

$$LUT_{indice} = [base\_address + offset], \quad (4.3)$$

où TC représente le nombre total de coefficients non nuls, CW représente le mot de code, *right_shift* représente la valeur du décalage de l'opération de décalage vers la droite et où les variables *start_TC*, *1stLevelOffset*, et *2ndLevelOffset*, sont fournies par :

$$start\_TC = \left\{ \begin{array}{ll} 2 & \text{si } TC < 7 \\ 7 & \text{sinon} \end{array} \right\} \quad (4.4)$$

$$1stLevelOffset = \left\{ \begin{array}{ll} 8 & \text{si } TC < 7 \\ 6 & \text{sinon} \end{array} \right\} \quad (4.5)$$

$$2ndLevelOffset = \left\{ \begin{array}{ll} 4 & \text{si } TC < 7 \\ 5 & \text{sinon} \end{array} \right\}. \quad (4.6)$$

Pour terminer cette sous-section, le mot de code (CW), qui est utilisé par les équations précédentes et servant d'indice pour l'une des deux (2) dimensions des tableaux 4.10 et 4.11, est dérivé de l'équation suivante :

$$CW = [bitstream\_chunk >> right\_shift] \quad (4.7)$$

où *bitstream_chunk* représente l'amalgame de bits extraient du train de bits et *right_shift* représente la valeur pour l'opération binaire de décalage vers la droite. L'obtention de la valeur de la variable *right_shift* est décrite dans la section 4.3.2.3.

4.3.2.3 Génération d'une table de contrôle

Cette sous-section présente le tableau 4.12 définissant une table de correspondance (LUT) de contrôle. Ce tableau de contrôle fut élaboré suite aux observations faites dans la sous-section 4.3.2.1 et sert à accéder les tableaux 4.10 et 4.11 présentés à la sous-section 4.3.2.2. La seule et unique dimension de cette LUT est indexée par la valeur TotalCoeff(*coeff_token*) - 1, ou bien $TC-1$. Chaque indice de la LUT fournit cinq (5) éléments. Le premier élément est le nombre de bits (*Nb Bits*) devant être extrait du train de bits de la trame vidéo compressée. Comme il est décrit dans la section précédente, l'amalgame de bits extraits du train de bits de la trame vidéo est aussi nommée *bitstream_chunk* (voir l'équation 4.7). Les quatre (4) derniers éléments de cette LUT représentent des seuils pour effectuer une opération de décalage vers la droite avec les valeurs 1, 2, 3, ou 4 respectivement. Ces seuils permettent de définir la variable *right_shift*, comme l'illustre l'équation suivante :

$$right\_shift = \left\{ \begin{array}{ll} 1 & \text{si } val_{decimale}(bitstream\_chunk) > Th_{RS1} \\ 2 & \text{si } val_{decimale}(bitstream\_chunk) > Th_{RS2} \\ 3 & \text{si } val_{decimale}(bitstream\_chunk) > Th_{RS3} \\ 4 & \text{si } val_{decimale}(bitstream\_chunk) > Th_{RS4} \\ 0 & \text{sinon} \end{array} \right\}. \tag{4.8}$$

Chaque dimension de la table de contrôle est représentée par la structure de l'extrait 4.2. Cette structure comporte six (6) champs qui, une fois combinés, occupent la taille d'un mot de 32 bits. Comme cette table de contrôle comporte 11 indexes (c.-à-d. pour $TC \in [2..12]$), la taille totale occupée en mémoire par celle-ci est de 44 octets.

Extrait 4.2 Code C de la structure utilisée par la table de contrôle

```
typedef struct _TOTAL_ZEROS_CTRL_STRUCT_ {
    tOCT_UINT32 u4NbBitsToRead     :  4;
    tOCT_UINT32 u2ShiftRight1Bits  :  2;
    tOCT_UINT32 u4ShiftRight2Bits  :  4;
    tOCT_UINT32 u5ShiftRight3Bits  :  5;
    tOCT_UINT32 u6ShiftRight4Bits  :  6;
    tOCT_UINT32 uPadding           : 11;
} tTOTAL_ZEROS_CTRL_STRUCT;
```

Tableau 4.12 Table de contrôle

indice (TC - 2)	Nb bits	Th_{RS1}	Th_{RS2}	Th_{RS3}	Th_{RS4}
0	6	3	7	23	63
1	6	1	7	23	63
2	5	3	11	31	31
3	5	1	11	31	31
4	6	1	3	7	63
5	6	1	3	7	47
6	6	1	3	7	31
7	6	1	3	7	15
8	5	1	3	7	31
9	4	1	7	7	15
10	4	1	3	7	15

4.3.2.4 Pseudo code de l'algorithme et exemples d'utilisation

L'extrait 4.3 présente le pseudo-code du nouvel algorithme pour extraire les mots de code *total_zeros*.

Extrait 4.3 Code C pour l'extraction de l'élément *total_zeros*

```
if((12 >= uiTotalCoeff) && (1 < uiTotalCoeff)) {
    t4BIT_TOTAL_ZEROS_STRUCT *ptsCavlcTotalZeros     = atsCavlcTotalZerosTc2ToTc6;
    tTOTAL_ZEROS_CTRL_STRUCT  uiCavlcTotalZerosCtrl = atsCavlcTotalZerosCtrl[uiTotalCoeff-1];
    unsigned int              uiTable1stLevelOffset = 8;
    unsigned int              uiTable2ndLevelOffset = 4;
    unsigned int              uiStartTableTC        = 2;
    unsigned int              uiRightShift          = 0;
    unsigned int              uiNbBitsToRead;
    unsigned int              uiBitstreamChunk;
    unsigned int              uiBaseAddress;
    unsigned int              uiCodeWord;

    /* STEP 1: bitstream chunk extraction */
    uiNbBitsToRead = atsCavlcTotalZerosCtrl[uiTotalCoeff - 1].u4NbBitsToRead;
    mSHOWBITS_VARIABLE(uiBitstreamChunk, uiNbBitsToRead, fi_ptsDecoderCtx);

    /* get code word (CW) */
    {
        /* get threshold from control table */
        unsigned int uiRightShift1Bits = uiCavlcTotalZerosCtrl.u2RightShift1Bits;
        unsigned int uiRightShift2Bits = uiCavlcTotalZerosCtrl.u4RightShift2Bits;
        unsigned int uiRightShift3Bits = uiCavlcTotalZerosCtrl.u5RightShift3Bits;
        unsigned int uiRightShift4Bits = uiCavlcTotalZerosCtrl.u6RightShift4Bits;

        /* STEP 2: set right_shift */
        uiRightShift = (uiBitstreamChunk > uiRightShift1Bits) ? 1 : uiRightShift;
        uiRightShift = (uiBitstreamChunk > uiRightShift2Bits) ? 2 : uiRightShift;
        uiRightShift = (uiBitstreamChunk > uiRightShift3Bits) ? 3 : uiRightShift;
        uiRightShift = (uiBitstreamChunk > uiRightShift4Bits) ? 4 : uiRightShift;

        /* STEP3: set code word (CW) */
        uiCodeWord = uiBitstreamChunk >> uiShiftRight;
    }

    /* STEP 4: select Table */
    if(6 < uiTotalCoeff) {
        ptsCavlcTotalZeros    = (t4BIT_TOTAL_ZEROS_STRUCT *)atsCavlcTotalZerosTc7ToTc12;
        uiTable1stLevelOffset = 6;
        uiTable2ndLevelOffset = 5;
        uiStartTableTC        = 7;
    }

    /* STEP 5: set base_address */
    uiBaseAddress = ((((uiTotalCoeff - uiStartTableTC) * uiTable1stLevelOffset) + uiCodeWord)
        * uiTable2ndLevelOffset);

    /* STEP 6: set total_zeros */
    uiTotalZeros = ptsCavlcTotalZeros[uiBaseAddress + uiRightShift].u4TotalZeros;

    /* STEP 7: set number of bits to flush from bitstream (VLC code size = num bits read - num
         unneeded bits) */
    uiNbBitsToFlush = uiNbBitsToRead - uiShiftRight;
}
```

Ce nouvel algorithme prend avantage du décodage arithmétique pour accélérer l'extraction des mots de code *total_zeros* sur les groupes provenant des tables de *TotalCoeff* allant de 2 à 12. Pour clarifier la nouvelle méthode de décodage proposée dans ce chapitre, les tableaux 4.13 et 4.14 présentent deux exemples dans le but d'illustrer le processus de l'algorithme.

Tableau 4.13 Exemple de la procédure de décodage lorsque TC = 3

Étape	Résultats de décodage	LUT utilisée	Indice
1	$num\_bits\_to\_read = 6$ $bitstream\_chunk = 001110$	Control	TC - 2 = 1
2	$TH_{RS3} > (ECW = 14) > TH_{RS2}$ $right\_shift = 2$	Control	TC - 2 = 1
3	$bitstream\_chunk = 0011$ $CW = 3$	-	-
4	$table\_1st\_level\_offset = 8$ $table\_2nd\_level\_offset = 4$ $start\_TC = 2$	-	-
5	$base\_address = (((3-2)\times 8)+3)\times 4 = 44$ $offset = 2$ $LUT_{indice} = 44+2$	-	-
6	$total\_zeros = 5$	TC = 2 à TC = 6	44 + 2 = 46
7	$num\_bits\_to\_flush = 6-2 = 4$	-	-

Tableau 4.14 Exemple de la procédure de décodage lorsque TC = 10

Étape	Résultats de décodage	LUT utilisée	Indice
1	$num\_bits\_to\_read = 5$ $bitstream\_chunk = 10010$	Control	TC - 2 = 8
2	$TH_{RS4} > (ECW = 18) > TH_{RS3}$ $right\_shift = 3$	Control	TC - 2 = 8
3	$bitstream\_chunk = 10$ $CW = 2$	-	-
4	$table\_1st\_level\_offset = 6$ $table\_2nd\_level\_offset = 5$ $start\_TC = 7$	-	-
5	$base\_address = (((10-7)\times 6)+2)\times 5 = 100$ $offset = 3$ $LUT_{indice} = 100+3$	-	-
6	$total\_zeros = 4$	TC = 7 à TC = 12	100 + 3 = 103
7	$num\_bits\_to\_flush = 5-3 = 2$	-	-

4.4 Conclusion

Dans ce chapitre, l'algorithme de décodage de l'entropie de type CAVLC de la spécification H.264 fut présenté. Ce type de décodage de l'entropie est composé de cinq (5) étapes pour extraire cinq éléments du train de bits d'une trame vidéo compressée. Le code de référence de la spécification H.264 utilise une méthode nommée TLSS pour extraire chacun des éléments de l'entropie de type CAVLC. L'exécution de ce type de méthode est considérée comme étant lente. Ce chapitre s'est spécifiquement concentré sur l'amélioration de la quatrième étape

du décodage de l'entropie de type CAVLC. Ce travail a donc proposé un nouvel algorithme pour extraire le nombre total de zéros à l'intérieur d'un bloc 4x4. Cette nouvelle méthode tire avantage du décodage arithmétique pour extraire l'élément *total_zeros*. La méthode proposée réutilise l'algorithme présenté par Kim *et al.* (2006) pour les tables de *TotalCoeff* 1, 13, 14, et 15, et propose un nouvel algorithme pour les tables de 2 à 12 inclusivement. Cette méthode utilise donc deux nouvelles tables de correspondance ainsi qu'une table de contrôle pour les accéder. Ces deux nouvelles tables de correspondance furent élaborées suite à l'analyse de la table VLC pour l'extraction du code *total_zeros* qui fut réordonnancé en s'inspirant du réarrangement proposé par l'article de Moon (2008). Le pseudo-code de l'algorithme ainsi que deux exemples d'utilisation de celui-ci furent présentés à la fin de ce chapitre. Les gains en performance de l'approche proposée en comparaison avec l'algorithme du code de référence de la spécification H.264 seront présentés au chapitre 5.

CHAPITRE 5

RÉSULTATS DE SIMULATIONS ET ANALYSE

Dans ce chapitre, nous présentons les résultats de simulations ainsi que leur analyse pour la solution du décodage en parallèle de l'implémentation du décodeur H.264 présentée au chapitre 3. De plus, nous présentons les résultats de simulations ainsi que leur analyse pour le nouvel algorithme proposé pour l'extraction du nombre total de zéros à l'intérieur d'un bloc 4x4 pour l'optimisation du décodage de l'entropie de type CAVLC présenté au chapitre 4.

Ce chapitre est divisé en quatre sections principales. À la section 5.1, nous décrivons les paramètres de test qui sont utilisés pour les résultats de simulations. Par la suite, à la section 5.2, nous présentons les résultats obtenus pour l'optimisation du décodage de l'entropie de type CAVLC, alors qu'à la section 5.3 nous présentons les résultats obtenus pour l'implémentation du décodeur parallèle H.264. Pour terminer, nous concluons sur les points discutés dans ce chapitre à la section 5.4.

5.1 Description des paramètres de test

Dans cette section nous présentons les paramètres de test qui sont utilisés pour les résultats de simulations à l'intérieur des sections 5.2 et 5.3. Tout d'abord, les simulations du présent travail utilisent sept (7) séquences vidéo non compressées. Ces séquences vidéo sont décrites dans le tableau 5.1. Trois (3) types de résolution sont utilisés pour ces séquences vidéo, soit la résolution 360p (c.-à-d. 640 x 360 pixels), la résolution NTSC (c.-à-d. 720 x 480 pixels), et la résolution HD 720p (c.-à-d. 1280 x 720 pixels).

Afin de générer la suite de séquences vidéo compressées utilisée pour les résultats de simulations à l'intérieur de ce chapitre, chacune de ces séquences vidéo présentées au tableau 5.1 fut compressée à l'aide de l'encodeur se nommant *x264* (x264, 2012) qui est disponible gratuitement. Nous avons utilisé tous les paramètres par défaut fournis par l'encodeur *x264* pour le profile de base (*baseline profile*) à l'exception de quelques paramètres spécifiques. Un résumé de ces paramètres spécifiques d'encodage utilisés pour chacune des trois résolutions est

Tableau 5.1 Description des séquences vidéo utilisées pour les simulations.

Séquence	**Description**
bad apple	Séquence vidéo basée sur un modèle 3D en mouvement et sur des jeux d'ombrages basés sur des silhouettes (Niconico, 2012).
big buck bunny	Séquence vidéo basée sur un film d'animation en 3D (Xiph.org, 2012).
controlled burn	Séquence vidéo basée sur une scène réelle provenant d'une séquence de tests redistribuable gratuitement (Xiph.org, 2012).
elephants dream	Séquence vidéo basée sur un film d'animation en 3D (Xiph.org, 2012).
moving zone plate	Séquence vidéo basée sur la fonction $sinc(x)$.
speed bag	Séquence vidéo basée sur une scène réelle provenant d'une séquence de tests redistribuable gratuitement (Xiph.org, 2012).
tractor	Séquence vidéo basée sur une scène réelle provenant d'une séquence de tests redistribuable gratuitement (Xiph.org, 2012).

fourni par le tableau 5.2. La présence du symbole X dans une case de ce tableau signifie que ce paramètre est utilisé pour cette résolution.

Tableau 5.2 Résumé des paramètres d'encodage pour chaque résolution.

Paramètres	**360p**	**NTSC**	**720p**
Une trame de référence (trame inter)	X	X	X
Utilisation du filtrage antibloc	X	X	X
Taille de paquets de 1400 octets	X	X	X
Débit d'encodage de 512 kb/s	X	X	
Débit d'encodage de 768 kb/s	X	X	
Débit d'encodage de 1000 kb/s	X	X	X
Débit d'encodage de 1500 kb/s			X
Débit d'encodage de 2000 kb/s	X	X	X
Débit d'encodage de 4000 kb/s			X

Tous les tests de simulations présentés dans ce chapitre furent exécutés sur le processeur multicoeurs DSP asynchrone *OCT1010* à l'aide de la plateforme logicielle *Vocallo*. Ce processeur possède 15 coeurs DSP. Quatre (4) de ces 15 coeurs DSP sont réservés exclusivement pour des processus propres à la plateforme logicielle *Vocallo*, ce qui laisse un total de 11 coeurs DSP disponibles afin d'exécuter l'implémentation du décodeur H.264 provenant de ce travail.

La plateforme logicielle *Vocallo* fut compilée et exécutée à l'aide du compilateur *octcc* et de l'environnement de développement *Octasic Opus Studio* provenant de la version 1.0.1.114. La taille de la mémoire externe de type *Mobile DDR*, qui fut utilisée lors des tests de simulations, est de 128 Mo.

5.2 Résultats obtenus pour l'optimisation du décodage de l'entropie de type CAVLC

Dans cette section, nous présentons les résultats obtenus à l'aide de la résolution 720p pour l'optimisation de l'entropie de type CAVLC, c.-à-d. pour le nouvel algorithme d'extraction du nombre total de zéros dans un bloc 4x4 présenté au chapitre 4, soit l'étape 4 du décodage de l'entropie de type CAVLC. L'implémentation du décodeur de ce travail n'utilise pas les algorithmes provenant du code source de référence fourni par la spécification H.264 pour les étapes 1, 2, 3, et 5, mais utilise plutôt des algorithmes plus optimaux provenant de la littérature (Iqbal et Henkel, 2009; Chen *et al.*, 2008).

Les vitesses d'exécution moyennes en trames par seconde (fps *pour frames per second*) et les accélérations du processus de décodage de l'entropie de type CAVLC pour des séquences vidéo utilisant la résolution 720p sont présentés dans les tableaux 5.3 et 5.4. Les résultats détaillés obtenus pour les résolutions 360p et NTSC sont présentés à l'annexe I.

Le tableau 5.3 contient les résultats pour les séquences vidéo ayant un débit d'encodage de 1000 kb/s et 1500 kb/s. Le tableau 5.4 contient les résultats pour les séquences vidéo ayant un débit d'encodage de 2000 kb/s et 4000 kb/s. À l'intérieur de ces tableaux, le paramètre R représente le nombre de trames par secondes (fps) que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique qui utilise l'algorithme fourni par le code de référence pour l'extraction de l'élément *total_zeros*. Le paramètre P représente le nombre de trames par secondes (fps) que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique qui utilise le nouvel algorithme proposé au chapitre 4 pour l'extraction de l'élément *total_zeros*. Le paramètre A représente l'accélération qui est calculée par l'équation suivante :

$$A = \frac{P}{R}. \tag{5.1}$$

Tableau 5.3 Vitesses d'exécution moyennes (fps) et accélérations moyennes du processus de décodage de l'entropie et de l'analyse syntaxique pour la résolution 720p utilisant des séquences vidéo à 1000 kb/s et 1500 kb/s.

Séquence	**1000 kb/s**			**1500 kb/s**		
	R (fps)	*P* (fps)	*A*	*R* (fps)	*P* (fps)	*A*
bad apple	45.830	46.528	1.015	40.202	41.154	1.024
big buck bunny	42.414	43.013	1.014	36.177	37.077	1.025
controlled burn	41.205	41.703	1.012	34.879	35.494	1.018
elephants dream	43.155	43.506	1.008	36.567	37.168	1.016
moving zone plate	42.393	42.621	1.005	35.228	35.565	1.010
speed bag	39.247	39.517	1.007	32.963	33.298	1.010
tractor	44.019	44.302	1.006	36.876	37.199	1.009
Moyenne	42.609	43.027	1.010	36.127	36.708	1.016

Tableau 5.4 Vitesses d'exécution moyennes (fps) et accélérations moyennes du processus de décodage de l'entropie et de l'analyse syntaxique pour la résolution 720p utilisant des séquences vidéo à 2000 kb/s et 4000 kb/s.

Séquence	**2000 kb/s**			**4000 kb/s**		
	R (fps)	*P* (fps)	*A*	*R* (fps)	*P* (fps)	*A*
bad apple	36.194	37.275	1.030	28.342	29.521	1.042
big buck bunny	31.696	32.820	1.035	22.115	23.743	1.074
controlled burn	30.474	31.148	1.022	20.783	21.614	1.040
elephants dream	31.931	32.773	1.026	21.402	22.332	1.043
moving zone plate	30.378	30.791	1.014	21.008	21.657	1.031
speed bag	28.569	28.907	1.012	19.044	19.485	1.023
tractor	31.896	32.274	1.012	21.409	21.904	1.023
Moyenne	31.591	32.284	1.022	22.015	22.894	1.039

La courbe d'accélération du nouvel algorithme d'extraction de l'élément *total_zeros* en fonction du débit d'encodage pour la résolution 720p est illustrée à la figure 5.1. Les données de cette figure proviennent des résultats obtenus pour les tableaux 5.3 et 5.4.

Il est possible de conclure à partir de la figure 5.1 que l'accélération du nouvel algorithme d'extraction de l'élément *total_zeros* par rapport à l'algorithme du code de référence est proportionnelle au débit d'encodage, c.-à-d. que l'accélération du nouvel algorithme augmente avec un débit d'encodage plus élevé. Cela s'explique par le fait que l'algorithme fourni par le code de référence possède une complexité $\Theta(n)$, où n représente le nombre maximal de coefficients non nuls que peut avoir le bloc 4x4 courant, soit une complexité linéaire, alors que

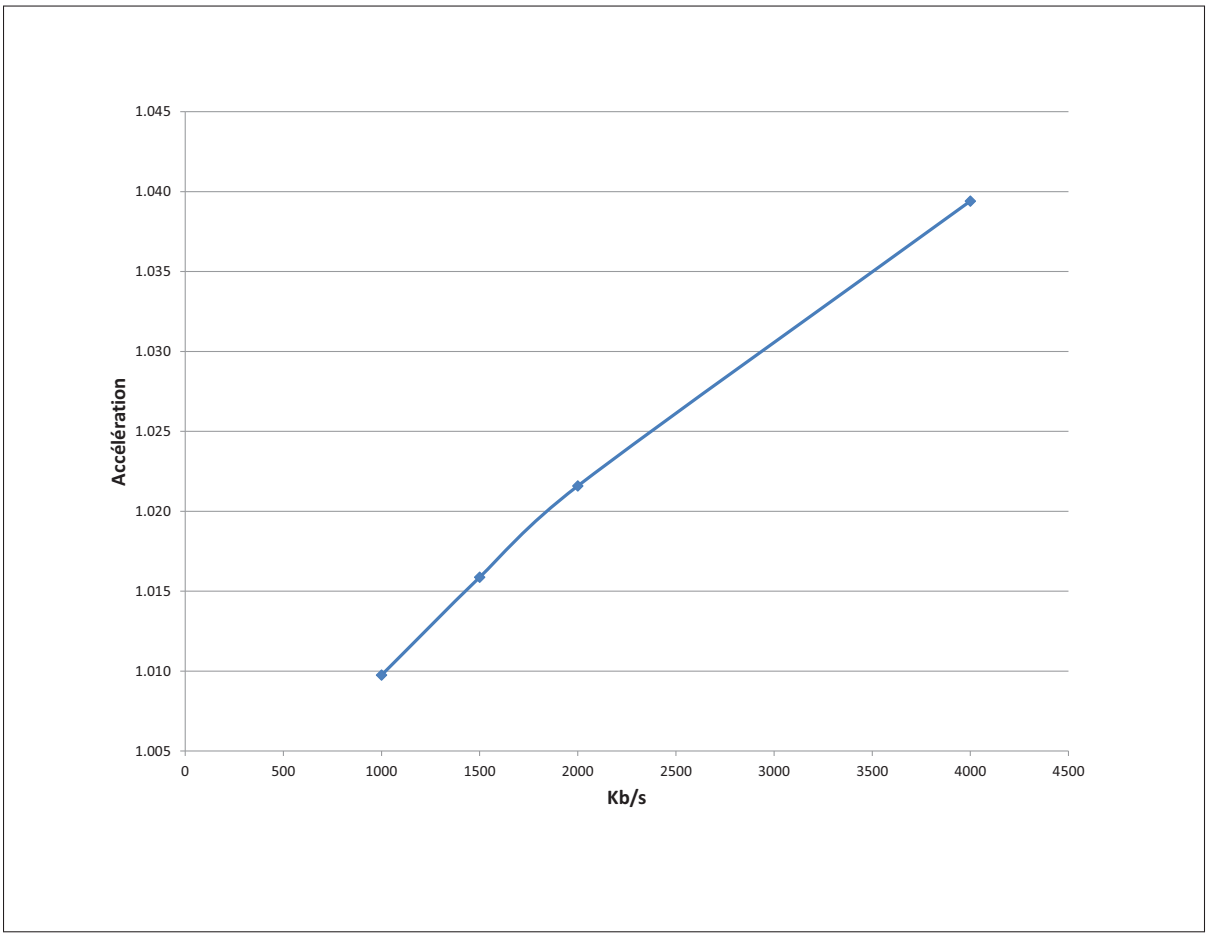


Figure 5.1 Courbe d'accélération du nouvel algorithme pour l'extraction de l'élément *total_zeros* pour la résolution 720p.

le nouvel algorithme proposé possède une complexité $\Theta(1)$, soit une complexité constante. Lorsque le débit d'encodage augmente, l'encodeur peut alors diminuer la compression de la séquence vidéo, c.-à-d. l'encodeur peut diminuer la distorsion utilisée à l'intérieur des trames vidéo constituant la séquence vidéo. Lorsqu'un bloc 4x4 n'est pas codé, cela implique que le processus du décodage de l'entropie et de l'analyse syntaxique n'a pas besoin de passer au travers des cinq (5) étapes du décodage de l'entropie de type CAVLC qui comprend, entre autres, l'extraction de l'élément *total_zeros*.

Bref, lorsque le débit d'encodage augmente, il y a moins de MB non codés, et plus de blocs 4x4 (avec un ou plusieurs coefficients nuls) qui vont bénéficier de l'algorithme proposé. L'accélération pourrait cependant atteindre un plateau à plus haut débit, car le nombre de coefficients

nuls diminue avec une augmentation du débit. Cependant, nos résultats de simulations n'illustrent pas cet effet avec les débits utilisés provenant des applications de vidéoconférence pour le domaine de la téléphonie mobile.

La courbe d'accélération du nouvel algorithme pour l'extraction de l'élément *total_zeros* en fonction du nombre de bits par MB seconde (c.-à-d. $b/(MB \times s)$) combinant les résolutions 360p, NTSC, et 720p, est présentée à la figure 5.2. La courbe démontre que l'accélération en fonction du nombre de bits par MB seconde a une forme logarithmique pour les débits utilisés par les tests de simulations de ce chapitre. Il est possible aussi d'observer sur cette figure que les résultats pour la résolution 720p se trouvent légèrement en dessous de la courbe. Cela s'explique par le fait que les macroblocs pour la résolution 720p possèdent une plus faible densité d'informations que pour les résolutions 360p et NTSC, ce qui entraîne un plus grand nombre de blocs 4x4 non codés.

Dans le reste de ce chapitre, tous les résultats de simulations concernant le processus du décodage de l'entropie et de l'analyse syntaxique utilisent le nouvel algorithme proposé par ce travail pour l'extraction de l'élément *total_zeros* faisant partie du décodage de l'entropie de type CAVLC.

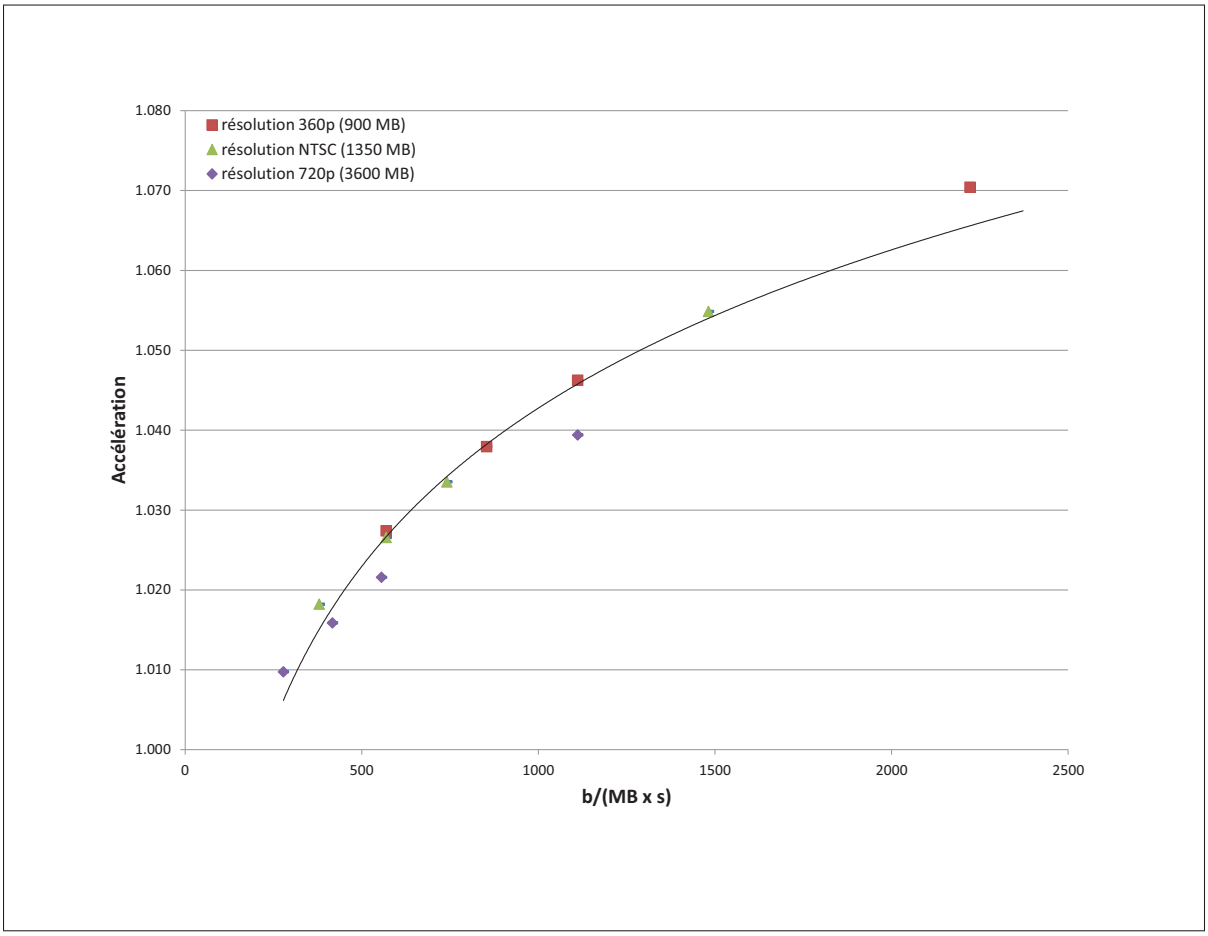


Figure 5.2 Courbe d'accélération du nouvel algorithme pour l'extraction de l'élément *total_zeros* en fonction du nombre de bits par MB seconde combinant les résolutions 360p, NTSC et 720p.

5.3 Résultats obtenus pour l'implémentation du décodeur H.264/MPEG4-AVC

Les résultats obtenus pour l'implémentation du décodeur H.264 utilisant les approches de parallélisation décrites au chapitre 3 pour la résolution 720p sont présentés dans cette section. Les résultats détaillés obtenus pour les résolutions 360p et NTSC sont présentés à l'annexe I. Tout d'abord, les résultats obtenus pour l'intervalle de notification de la position d'un indice de MB pour la synchronisation avec dépendances de données sont présentés à la sous-section 5.3.1. Par la suite, les résultats obtenus sur la répartition du nombre de coeurs DSP pour chacun des deux processus du décodeur H.264 sont présentés à la sous-section 5.3.2. Pour terminer, les résultats obtenus sur l'accélération du décodeur H.264 sont présentés à la sous-section 5.3.3.

5.3.1 Intervalle de notification de la position d'un indice de MB

Le sommaire des vitesses d'exécution moyennes en fps du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour des séquences vidéo utilisant la résolution 720p pour différents débits d'encodage est présenté au tableau 5.5. Le concept d'intervalle de notification en macrobloc fut présenté au chapitre 3 (section 3.2.2.3).

Tableau 5.5 Sommaire des vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution 720p.

Sommaire	**1 MB**	**2 MB**	**4 MB**	**5 MB**	**6 MB**	**8 MB**	**12 MB**	**16 MB**
512 kb/s	37.144	37.648	37.204	36.781	36.453	35.031	29.151	23.953
768 kb/s	36.464	36.906	36.457	36.036	35.738	34.356	28.604	23.499
1000 kb/s	35.959	36.375	35.931	35.512	35.202	33.834	28.170	23.143
2000 kb/s	34.462	34.809	34.359	33.944	33.637	32.345	26.852	22.042
Moyenne	36.007	36.434	35.988	35.568	35.257	33.891	28.194	23.159

Ces résultats démontrent que l'intervalle de notification de la position d'un indice de MB pour la synchronisation avec dépendances de données a une influence directe sur la vitesse d'exécution moyenne pour le processus de la reconstruction et du filtrage antibloc du décodeur H.264. En effet, il est possible d'observer dans ce tableau que la vitesse d'exécution moyenne diminue lorsque l'intervalle de notification de la position d'un indice de MB augmente.

La courbe de la vitesse d'exécution moyenne en fps du processus de la reconstruction et du filtrage antibloc en fonction de la taille de l'intervalle de notification en macrobloc utilisée pour la synchronisation de tâches avec dépendances de données pour la résolution 720p est illustrée à la figure 5.3. Les données de cette figure proviennent des résultats obtenus pour le tableau 5.5.

L'hypothèse de départ était que l'intervalle de notification optimale de la position d'un indice de MB serait de 1 macrobloc. Par contre, la figure 5.3 démontre que l'intervalle optimal de notification de la position d'un indice de MB sur le processeur DSP *OCT1010* est de 2 macroblocs. Ce même phénomène est observé pour la résolution NTSC, alors qu'il n'est pas observé

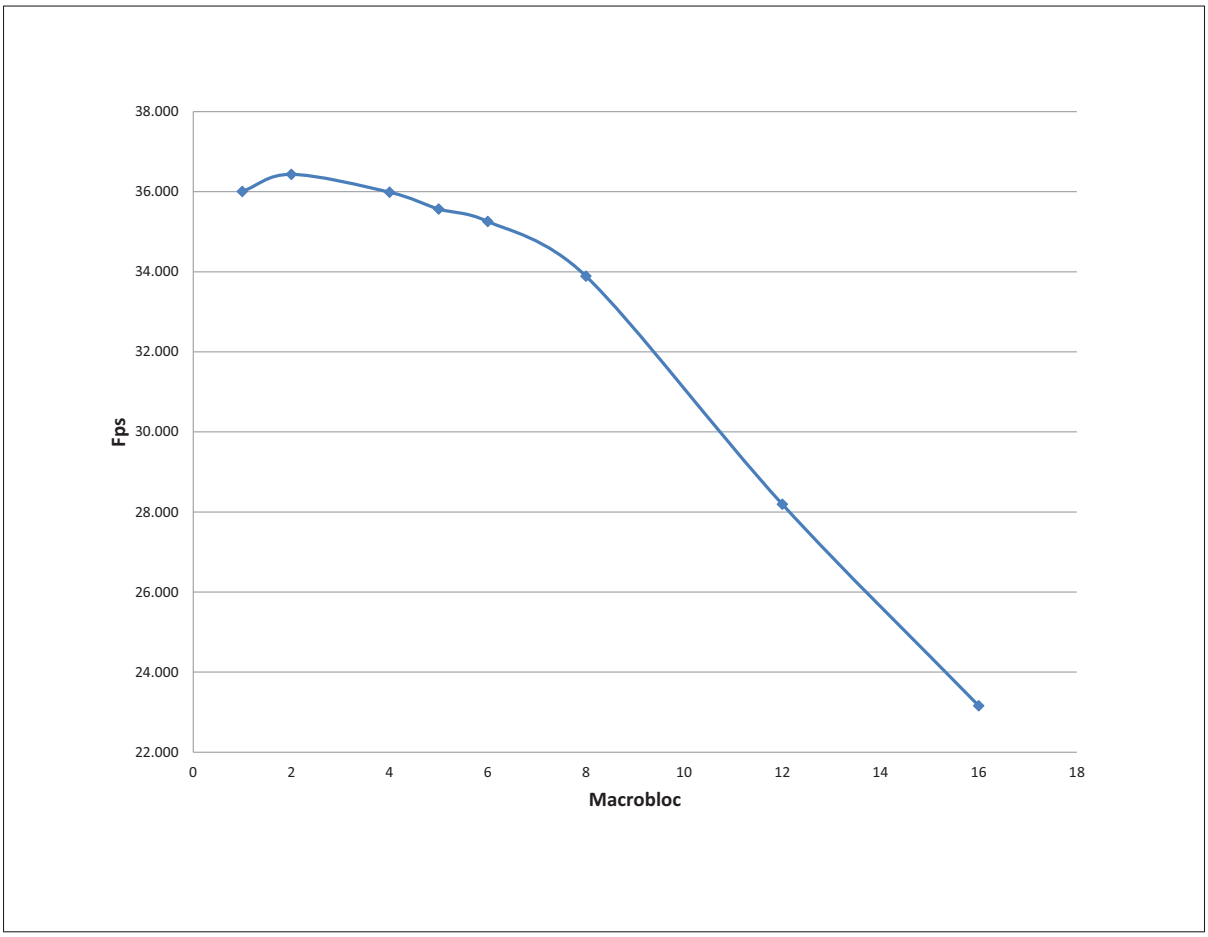


Figure 5.3 Courbe de la vitesse d'exécution moyenne du processus de la reconstruction et du filtrage antibloc en fonction de la taille de l'intervalle de notification en macrobloc pour la résolution 720p.

pour la résolution 360p (voir les résultats détaillés à l'annexe I). Cela peut s'expliquer par la nature du processeur DSP *OCT1010* qui est basé sur une architecture Von Neumann (voir le chapitre 1). Avec un très grand nombre de données transitant sur le bus mémoire, ce dernier perd légèrement de son efficacité avec un trop grand nombre de notifications entre coeurs DSP.

De plus, nous remarquons que pour la résolution NTSC, l'intervalle de 2 MB est légèrement meilleur que l'intervalle de 1 MB, donc très près du point de rupture. Pour la résolution 720p la différence est plus marquée. Cela présage qu'à plus haute résolution, il est possible qu'un intervalle de 2 MB ne soit même plus optimal. Pour la méthodologie, il est important de remarquer que bien que l'on affiche la moyenne seulement, la tendance s'observe séquence par séquence et débit par débit (c.-à-d. pour chacun l'optimal est le même). Si la tendance n'était pas si forte,

il serait beaucoup moins utile de faire une moyenne sur des données non consistantes et tirer des conclusions sur la moyenne.

Dans le reste de ce chapitre, tous les résultats de simulations pour la synchronisation avec dépendances de données (c.-à-d. pour le processus du décodage de l'entropie et de l'analyse syntaxique) utilisent un intervalle de 2 MB pour la notification de la position d'un indice de macrobloc.

5.3.2 Répartition du nombre de coeurs DSP alloués pour chaque processus

Cette section a pour objectif de déterminer les configurations parallèles optimales pour les résolutions 360p, NTSC et 720p. Ces configurations sont représentées par les combinaisons x/y, où x représente le nombre de coeurs DSP pour le processus du décodage de l'entropie et de l'analyse syntaxique et y représente le nombre de coeurs DSP pour le processus de la reconstruction et du filtrage antibloc.

Le sommaire des vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant différents débits d'encodage pour la résolution 720p et où le décodeur utilise de 1 à 9 coeurs DSP est présenté dans les tableaux 5.6, 5.7, et 5.8. À l'intérieur de ces tableaux, le paramètre f_1 représente le nombre de trames par seconde que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique, et le paramètre f_2 représente le nombre de trames par seconde que peut traiter le processus de la reconstruction et du filtrage antibloc. Chacun des processus f_1 et f_2 utilise le nombre de coeurs indiqué dans la colonne des tables. Notons aussi que pour le processus f_1, nous n'avons pas indiqué les vitesses pour des nombres élevés de coeurs puisque ce processus, disposant de quelques coeurs, est déjà plus rapide que le processus f_2 même lorsque ce dernier dispose de 9 coeurs. La dernière ligne de ces tableaux représente le minimum des temps d'exécution moyens de chacune des colonnes et ces résultats sont utilisés à l'intérieur du tableau 5.9. Les résultats détaillés obtenus pour les résolutions 360p et NTSC sont présentés à l'annexe I (les accélérations sont plus élevées pour ces résolutions).

Les courbes des vitesse d'exécution moyennes pour la résolution 720p utilisant différents nombre de coeurs DSP pour le processus du décodage de l'entropie et de l'analyse syntaxique

Tableau 5.6 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	43.027	6.679	67.115	11.973	80.640	17.600
1500 kb/s	36.708	6.535	62.118	11.725	76.164	17.236
2000 kb/s	32.284	6.434	56.016	11.553	73.642	16.986
4000 kb/s	22.894	6.130	41.970	11.028	57.704	16.220
minimum	22.894	6.130	41.970	11.028	57.704	16.220

Tableau 5.7 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	85.335	21.774	87.043	27.350	-	30.510
1500 kb/s	90.753	21.365	95.775	26.762	-	29.880
2000 kb/s	84.137	21.048	94.885	26.357	-	29.457
4000 kb/s	70.898	20.108	80.898	25.164	-	28.147
minimum	70.898	20.108	80.898	25.164	-	28.147

et pour le processus de la reconstruction et du filtrage antibloc sont présentées à la figure 5.4. Les données de cette figure proviennent des résultats obtenus pour les tableaux 5.6, 5.7 et 5.8. Il est possible d'observer à l'aide des courbes de cette figure que les deux processus du décodage H.264 de ce travail, soit le processus du décodage de l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc, sont très débalancés au niveau de la complexité.

Le minimum des vitesses d'exécution moyennes (fps) pour différentes configurations de coeurs DSP pour des séquences vidéo utilisant les résolutions 360p, NTSC et 720p, sont présentés dans le tableau 5.9. Les cellules grises à l'intérieur de ce tableau indiquent les vitesses moyennes d'exécution ne rencontrant pas la restriction de *temps réel* pour le domaine de la vidéoconférence utilisant une communication à 30fps. Par la suite, les résultats en gras indiquent les vitesses moyennes d'exécution optimales rencontrant la restriction de *temps réel*.

Il est possible de conclure, à la lecture du tableau 5.9, que la résolution 360p rencontre le critère de temps réel à partir de la configuration parallèle $1/2$, soit lors de l'utilisation de 3 coeurs DSP.

Tableau 5.8 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	-	33.835	-	37.649	-	40.251
1500 kb/s	-	33.102	-	36.906	-	39.490
2000 kb/s	-	32.579	-	36.375	-	38.883
4000 kb/s	-	31.127	-	34.809	-	37.251
minimum	-	31.127	-	34.809	-	37.251

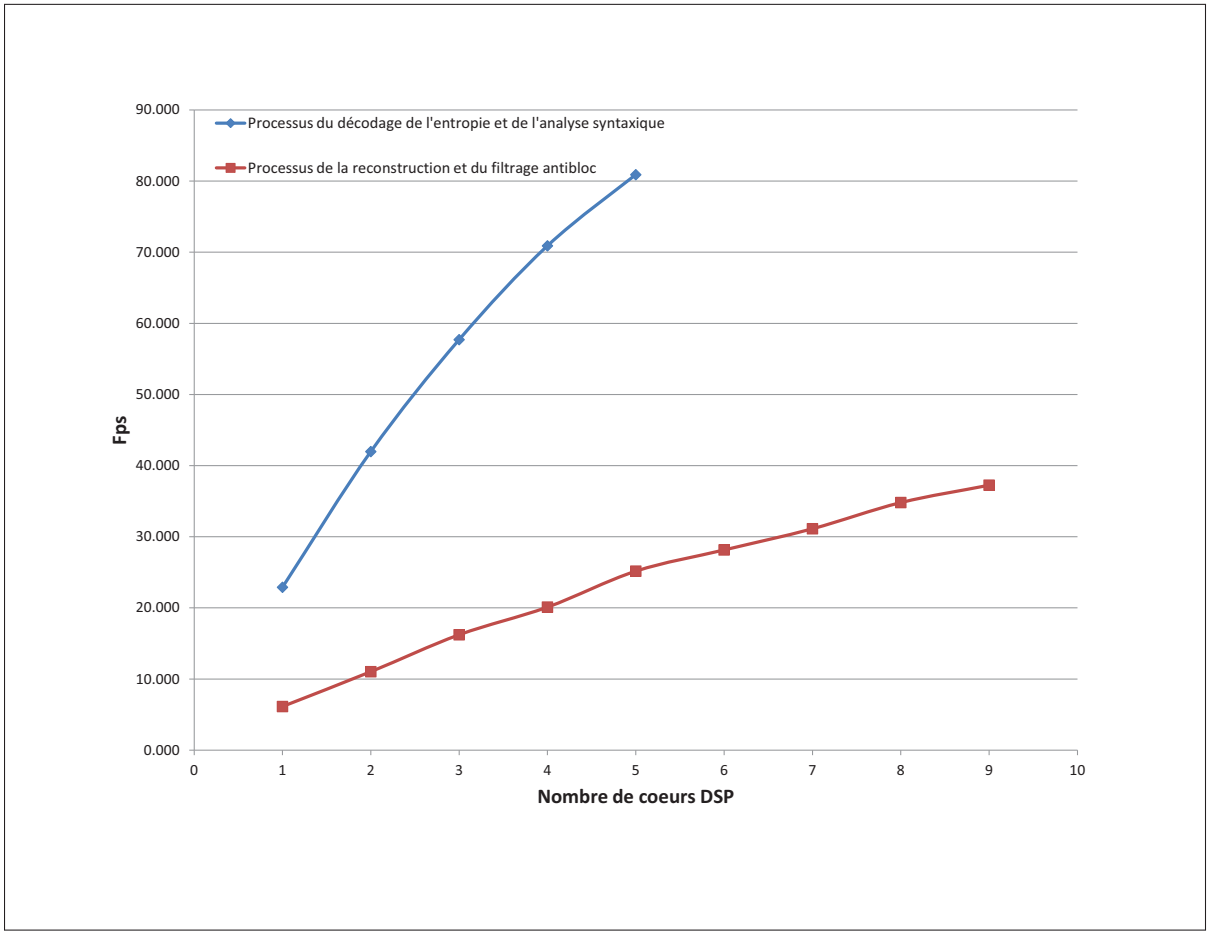


Figure 5.4 Courbes des vitesses d'exécution moyennes pour la résolution 720p utilisant différents nombres de coeurs DSP pour le processus du décodage de l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc.

La résolution NTSC rencontre le critère de temps réel à partir de la configuration parallèle $1/3$, soit lors de l'utilisation de 4 coeurs DSP. Pour terminer, la résolution 720p rencontre le critère de temps réel à partir de la configuration parallèle $2/7$, soit lors de l'utilisation de 2 coeurs

Tableau 5.9 Vitesses d'exécution moyennes (fps) pour différentes configurations de coeurs DSP pour des séquences vidéo utilisant les résolutions 360p, NTSC et 720p

Nb Coeurs	**360p**		**NTSC**		**720p**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1	**61.909**	22.711	**51.579**	16.087	22.894	6.130
2	105.181	**37.985**	88.626	27.872	**41.970**	11.028
3	162.288	53.818	117.778	**40.039**	57.704	16.220
4	162.147	68.744	134.559	49.629	70.898	20.108
5	167.161	78.277	135.372	59.190	80.898	25.164
6	-	89.026	-	67.133	-	28.147
7	-	96.249	-	72.520	-	**31.127**
8	-	106.904	-	80.531	-	34.809
9	-	106.258	-	82.027	-	37.251

DSP pour le processus du décodage de l'entropie et de l'analyse syntaxique et de 7 coeurs DSP pour le processus de la reconstruction et du filtrage antibloc pour un total de 9 coeurs DSP.

5.3.3 Résultats d'accélérations du décodeur

Le sommaire des vitesses d'exécution moyennes en fps et de l'accélération du décodeur H.264 pour différentes configurations de coeurs DSP pour la résolution 720p est présenté dans les tableaux 5.10, 5.11, et 5.12. À l'intérieur de ces tableaux, le paramètre f_1 représente le nombre de trames par secondes (fps) que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique, le paramètre f_2 représente le nombre de trames par secondes (fps) que peut traiter le processus de la reconstruction et du filtrage antibloc, le paramètre S représente le nombre de trames par secondes (fps) que peut traiter le système global du décodage H.264, c.-à-d. $1/(1/f_1 + 1/f_2)$ pour l'approche séquentielle et $min(f_1, f_2)$ pour les autres cas parallèles, et le paramètre A représente l'accélération qui est fournie par l'équation suivante :

$$A = \frac{TempsSystèmeRéférence}{TempsSystèmeNouveau} = \frac{FpsSystèmeNouveau}{FpsSystèmeRéférence}, \tag{5.2}$$

dans lequel *Nouveau* est un système proposé à l'aide d'une configuration parallèle de coeurs DSP et *Référence* est le système de départ, soit le système avec une configuration séquentielle n'utilisant qu'un seul coeur DSP.

Le tableau 5.10 contient les résultats pour la configuration séquentielle n'utilisant qu'un seul coeur DSP et pour la configuration parallèle utilisant 2 coeurs DSP, c.-à-d. la configuration parallèle 1/1. Le tableau 5.11 contient les résultats pour la configuration parallèle utilisant 5 coeurs DSP, c.-à-d. la configuration parallèle 2/3, et contient les résultats pour la configuration parallèle utilisant 7 coeurs DSP, c.-à-d. la configuration parallèle 2/5. Pour terminer, le tableau 5.12 contient les résultats pour la configuration parallèle utilisant 9 coeurs DSP, c.-à-d. la configuration parallèle 2/7, et contient les résultats pour la configuration parallèle utilisant 11 coeurs DSP, c.-à-d. la configuration parallèle 2/9.

L'histogramme des vitesses d'exécution moyennes pour différentes configurations de coeurs DSP pour la résolution 720p est illustré à la figure 5.5. Par la suite, la courbe d'accélération en fonction des différentes configurations de coeurs DSP pour la résolution 720p est illustrée à la

Tableau 5.10 Vitesses d'exécution moyennes (fps) et accélération pour des séquences vidéo utilisant la résolution 720p où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Sommaire	**Séquentielle 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
1000 kb/s	43.027	6.679	5.767	1.000	43.027	6.679	6.679	1.158
1500 kb/s	36.708	6.535	5.535	1.000	36.708	6.535	6.535	1.181
2000 kb/s	32.284	6.434	5.354	1.000	32.284	6.434	6.434	1.202
4000 kb/s	22.894	6.130	4.828	1.000	22.894	6.130	6.130	1.270
Moyenne	33.728	6.444	5.371	1.000	33.728	6.444	6.444	1.203

Tableau 5.11 Vitesses d'exécution moyennes (fps) et accélération pour des séquences vidéo utilisant la résolution 720p où le décodeur utilise une configuration de coeurs DSP parallèle 2/3 et parallèle 2/5.

Sommaire	**Parallèle 2/3 coeurs DSP**				**Parallèle 2/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
1000 kb/s	67.115	17.600	17.600	3.052	67.115	27.350	27.350	4.743
1500 kb/s	62.118	17.236	17.236	3.114	62.118	26.762	26.762	4.835
2000 kb/s	56.016	16.986	16.986	3.173	56.016	26.357	26.357	4.923
4000 kb/s	41.970	16.220	16.220	3.359	41.970	25.164	25.164	5.212
Moyenne	56.805	17.011	17.011	3.175	56.805	26.408	26.408	4.928

figure 5.6. Pour terminer, la courbe d'accélération en fonction du nombre de coeurs DSP pour la résolution 720p est illustrée à la figure 5.7. Les données de ces trois (3) figures proviennent des résultats obtenus pour les tableaux 5.10, 5.11, et 5.12.

Il est possible de conclure en observant les figures 5.5 et 5.6 que le nombre de trames par seconde (fps) que peut traiter le nouveau système global du décodage H.264 est supérieur au système global du décodage H.264 de référence n'utilisant qu'un seul coeur DSP. De plus, il est possible d'observer à l'aide des tableaux 5.10, 5.11, et 5.12, que l'accélération augmente avec le débit. Pour terminer, il est possible de conclure en observant la figure 5.7 que le gain d'accélération du décodeur H.264 augmente de façon linéaire lorsque le nombre de coeurs DSP utilisé augmente, ce qui concorde avec ce qui était attendu. Par contre, ce gain d'accélération n'est pas directement proportionnel au nombre de coeurs DSP utilisés, car, comme il fut démontré à la section 5.3.2, les deux processus du décodage H.264 de ce travail sont très débalancés au

Tableau 5.12 Vitesses d'exécution moyennes (fps) et accélération pour des séquences vidéo utilisant la résolution 720p où le décodeur utilise une configuration de coeurs DSP parallèle 2/7 et parallèle 2/9.

Sommaire	**Parallèle 2/7 coeurs DSP**				**Parallèle 2/9 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
1000 kb/s	67.115	33.835	33.835	5.867	67.115	40.251	40.251	6.980
1500 kb/s	62.118	33.102	33.102	5.981	62.118	39.490	39.490	7.135
2000 kb/s	56.016	32.579	32.579	6.085	56.016	38.883	38.883	7.262
4000 kb/s	41.970	31.127	31.127	6.447	41.970	37.251	37.251	7.715
Moyenne	56.805	32.661	32.661	6.095	56.805	38.969	38.969	7.273

niveau de la complexité. Les mêmes conclusions sont observées pour les résolutions 360p et NTSC.

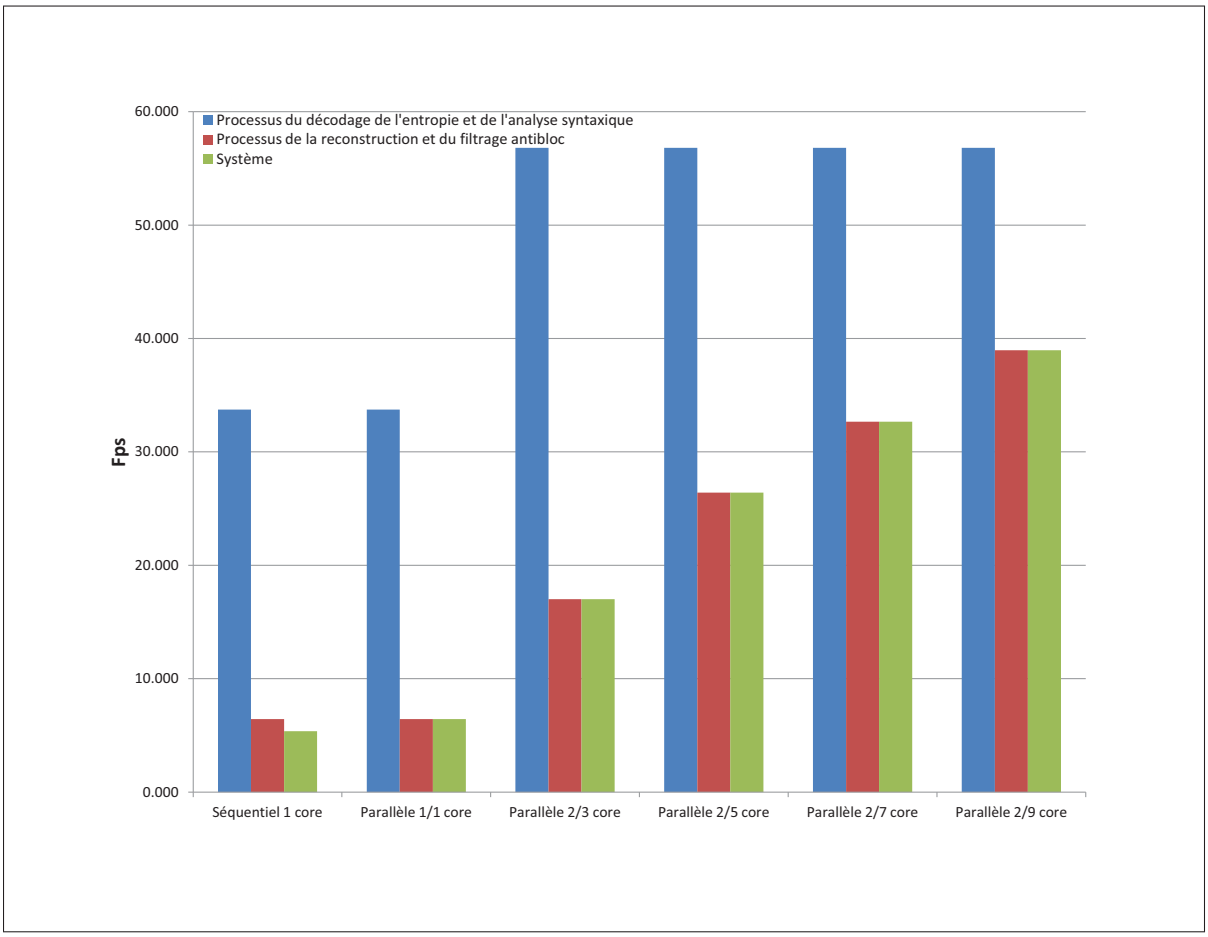


Figure 5.5 Histogramme des vitesses d'exécution moyennes pour différentes configurations de coeurs DSP pour la résolution 720p.

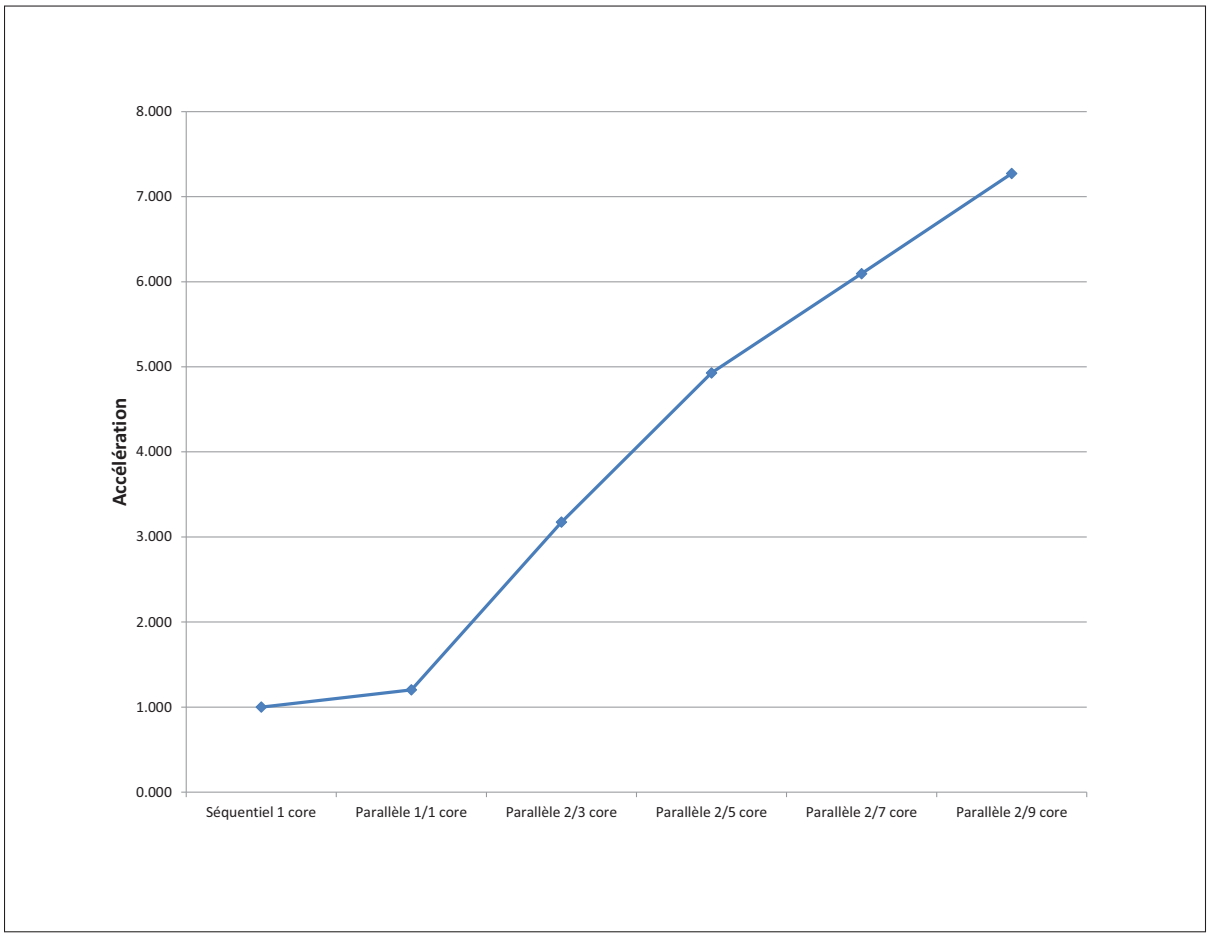


Figure 5.6 Courbe d'accélération en fonction de différentes configurations de coeurs DSP pour la résolution 720p.

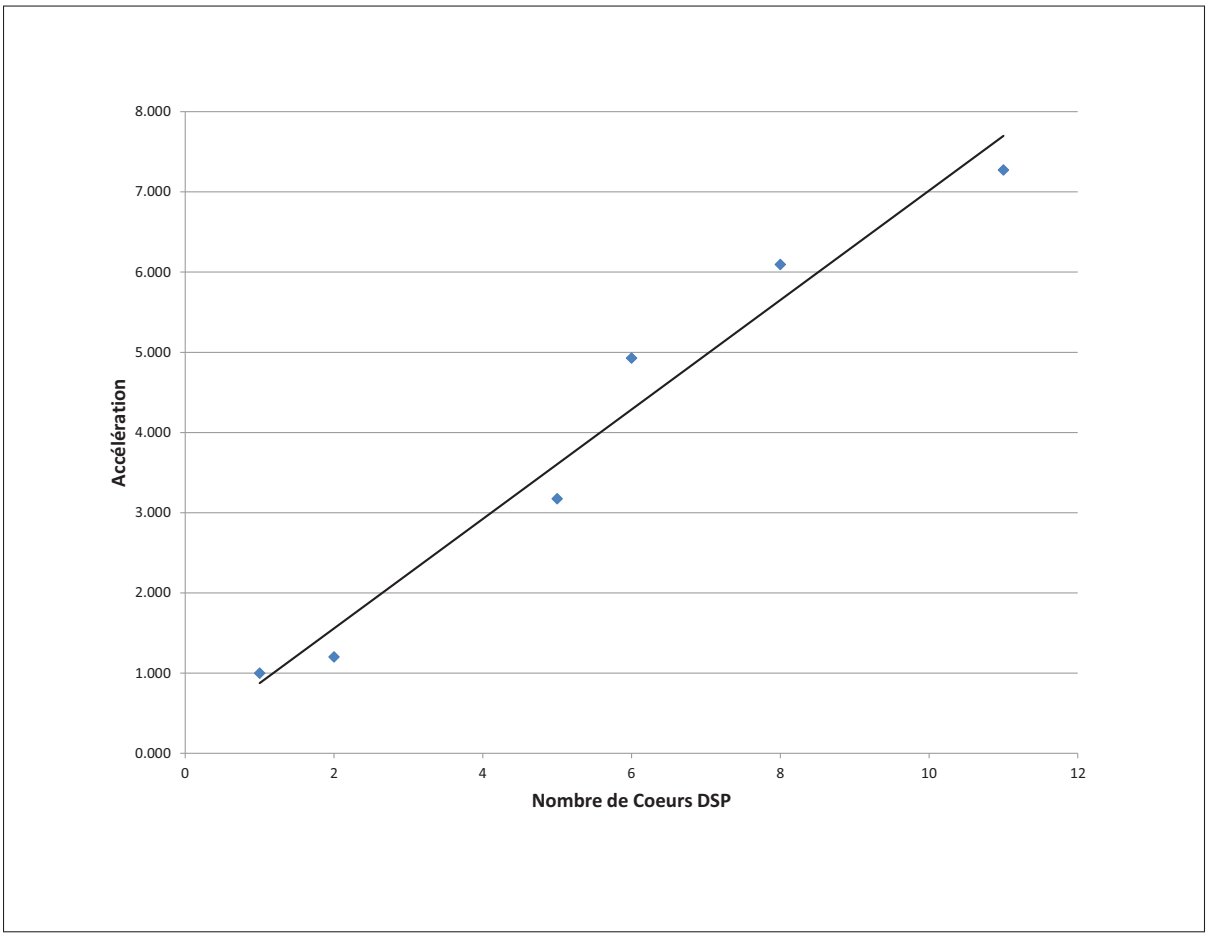


Figure 5.7 Courbe d'accélération en fonction du nombre de coeurs DSP pour la résolution 720p.

5.4 Conclusion

Dans ce chapitre, les résultats de simulations de l'implémentation du décodeur H.264 utilisant la solution de décodage en parallèle proposée au chapitre 3 furent présentés ainsi que les résultats de simulations du nouvel algorithme pour l'extraction de l'élément *total_zeros* pour un bloc 4x4 à l'intérieur du processus de décodage de l'entropie et de l'analyse syntaxique proposée au chapitre 4. Suite à l'analyse des résultats présentés dans ce chapitre, il fut démontré que l'algorithme proposé pour l'optimisation du décodage de l'entropie de type CAVLC est plus performant que l'algorithme fournit par le code de référence de la spécification H.264. Les gains peuvent atteindre jusqu'à 7% pour certaines combinaisons de séquence, débit et résolution. Par la suite, les gains d'accélération pour la solution du décodage en parallèle appliquée

pour le décodeur H.264 furent présentés et analysés. L'implémentation proposée du décodeur permet d'atteindre la contrainte de *temps réel* pour le domaine de la vidéoconférence, qui utilise une communication à 30fps, en utilisant 3 coeurs DSP pour la résolution 360p, 4 coeurs DSP pour la résolution NTSC et 9 coeurs DSP pour la résolution 720p. Pour terminer, il fut observé que l'intervalle de notification optimal pour l'indice de MB pour le processus de la reconstruction et du filtrage antibloc (c.-à-d. pour le processus utilisant une synchronisation avec dépendances de données) sur le processeur DSP *OCT1010* était de 2 macroblocs pour les résolutions NTSC et 720p, et de 1 macrobloc pour une plus petite résolution, soit la résolution 360p.

CONCLUSION GÉNÉRALE

Dans ce mémoire, nous avons présenté les différents concepts et aspects concernant le décodage de séquences vidéos à l'aide du standard H.264/MPEG-4 AVC sur une plateforme DSP asynchrone multicoeurs dédiée à la classe d'applications de la téléphonie mobile. Une revue de littérature a ensuite été effectuée sur les différentes méthodes de parallélisation du décodage avec le standard H.264/MPEG-4 AVC. Deux méthodes furent présentées, soit la méthode fondée sur le partitionnement de l'algorithme à l'aide de la fonctionnalité ainsi que la méthode fondée sur le partitionnement de l'algorithme à l'aide des données.

Par la suite, nous avons présenté notre nouvelle solution proposée au problème du décodage parallèle H.264/MPEG-4 AVC avec profil de base. Cette nouvelle solution tire avantage de l'architecture du processeur DSP asynchrone multicoeurs *OCT1010*. La solution que nous avons proposée se démarque des autres implémentations présentées dans la littérature principalement parce qu'il s'agit de la première implémentation utilisant une approche hybride sur un processeur DSP multicoeurs. En effet, la solution proposée utilise une approche basée sur la fonctionnalité où chacune des fonctionnalités utilise une approche basée sur les données. L'implémentation du décodeur utilise deux fonctionnalités, soit le processus du décodage de l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc. De plus, pour répondre aux contraintes de *temps réel* pour le domaine de la téléphonie mobile, cette solution propose de pipeliner ces deux fonctionnalités.

Cette solution proposée pour l'implémentation du décodeur H.264/MPEG-4 AVC supporte aussi plusieurs concepts d'extensibilité. Cette solution utilise tout d'abord un mécanisme d'abstraction de la résolution pour rendre le modèle mémoire extensible pour n'importe quel type de résolutions pour un processeur DSP avec des ressources mémoires locales limitées. Par la suite, cette solution utilise un modèle de synchronisation et d'intercommunication générique applicable autant pour les tâches indépendantes que pour les tâches avec dépendances de données tout en restant extensible par rapport au nombre de coeurs DSP disponibles.

Par ailleurs, nous avons aussi présenté une solution pour l'optimisation du décodage de l'entropie de type CAVLC pour la spécification H.264/MPEG-4 AVC. Un nouvel algorithme est proposé pour améliorer le temps d'exécution de la quatrième étape du décodage de l'entropie de type CAVLC, soit l'extraction du nombre total de zéros à l'intérieur d'un bloc 4x4. Cette nouvelle solution tire avantage du décodage arithmétique pour extraire l'élément *total_zeros* et possède une complexité d'exécution $\Theta(1)$, soit une complexité constante, comparativement à une complexité d'exécution $\Theta(n)$, soit une complexité linéaire, pour l'algorithme fournit par le code de référence de la spécification H.264/MPEG-4 AVC.

Les résultats de simulations ont de plus démontré que ce nouvel algorithme est plus performant que l'algorithme fournit par le code de référence. Le processus du décodage de l'entropie et de l'analyse syntaxique possède une amélioration moyenne du temps moyen d'exécution de l'ordre de 1% pour des séquences vidéos utilisant un débit d'encodage de 1000kbps pour la résolution HD 720p. Cette amélioration moyenne croît proportionnellement avec le débit d'encodage. En effet, l'amélioration moyenne du temps moyen d'exécution du processus du décodage de l'entropie et de l'analyse syntaxique est de l'ordre de 4% pour des séquences vidéos utilisant un débit d'encodage de 4000kbps pour la résolution HD 720p.

Pour terminer, nous avons présenté et analysé les résultats de simulations de la solution proposée au décodage parallèle pour l'implémentation du décodeur H.264/MPEG-4 AVC avec profil de base. Il fut démontré que l'implémentation du décodeur sur le processeur DSP atteint les contraintes de *temps réel* pour le domaine de la téléphonie mobile, soit pour les applications de vidéoconférences. En effet, la solution proposée par ce mémoire appliquée sur 11 coeurs DSP pour la résolution HD 720p possède un temps d'exécution représentant 130% des contraintes de *temps réel* comparativement à 18% pour une implémentation séquentielle n'utilisant qu'un seul coeur DSP, ce qui représente un gain d'accélération moyen de 7.3 pour le temps de décodage.

Les objectifs de ce mémoire furent donc atteints, soit en premier lieu d'optimiser le temps d'exécution du décodage de l'entropie de type CAVLC pour la spécification H.264/MPEG-4 AVC en proposant une solution originale qui améliore le temps d'exécution de l'une de ces

étapes. En deuxième lieu, proposer une nouvelle solution au décodage en parallèle H.264/MPEG-4 AVC avec profil de base sur un processeur DSP asynchrone multicoeurs pour le domaine de la téléphonie mobile qui supporte des tailles d'images allant jusqu'à la résolution HD 720p tout en respectant une contrainte de *temps réel* pour le débit de trame typique pour les applications de vidéoconférence en télécommunication.

ANNEXE I

RÉSULTATS DÉTAILLÉS DE SIMULATIONS

Tous les résultats de simulations qui n'ont pas été présentés au chapitre 5 se retrouvent dans cette annexe. Cette annexe est divisée en quatre sections principales. À la section 1, nous présentons les résultats détaillés pour l'optimisation du décodage de l'entropie de type CAVLC. Par la suite, à la section 2, nous présentons les résultats détaillés pour l'intervalle de notification de la position d'un indice de MB pour la synchronisation avec dépendances de données. Pour terminer, à la section 4, nous présentons les résultats détaillés de l'accélération du décodeur H.264 à l'aide des méthodes de parallélisation.

1 Résultats obtenus pour l'optimisation du décodage de l'entropie de type CAVLC

1.1 Résolution 360p

Les vitesses d'exécution moyennes en fps et les accélérations du processus de décodage de l'entropie de type CAVLC pour des séquences vidéo utilisant la résolution 360p sont présentées dans les tableaux I-1 et I-2. Le tableau I-1 contient les résultats pour les séquences vidéo ayant un débit d'encodage de 512 kb/s et 768 kb/s. Le tableau I-2 contient les résultats pour les séquences vidéo ayant un débit d'encodage de 1000 kb/s et 2000 kb/s.

Tableau-A I-1 Vitesses d'exécution moyennes (fps) et accélération moyenne du processus de décodage de l'entropie et de l'analyse syntaxique pour la résolution 360p utilisant des séquences vidéo encodées avec un débit de 512 kb/s et 768 kb/s.

Séquence	**512 kb/s**			**768 kb/s**		
	R (fps)	P (fps)	A	R (fps)	P (fps)	A
bad apple	139.503	144.124	1.033	117.461	122.059	1.039
big buck bunny	124.438	129.568	1.041	101.106	107.297	1.061
controlled burn	120.266	124.671	1.037	97.004	101.841	1.050
elephants dream	120.771	124.109	1.028	97.142	101.167	1.041
moving zone plate	117.685	120.028	1.020	94.561	97.233	1.028
speed bag	110.098	111.983	1.017	87.500	89.485	1.023
tractor	120.887	122.841	1.016	97.325	99.563	1.023
Moyenne	121.950	125.332	1.027	98.871	102.664	1.038

Tableau-A I-2 Vitesses d'exécution moyennes (fps) et accélération moyenne du processus de décodage de l'entropie et de l'analyse syntaxique pour la résolution 360p utilisant des séquences vidéo encodées avec un débit de 1000 kb/s et 2000 kb/s.

Séquence	**1000 kb/s**			**2000 kb/s**		
	R (fps)	P (fps)	A	R (fps)	P (fps)	A
bad apple	104.425	108.884	1.043	86.392	90.223	1.044
big buck bunny	86.925	93.618	1.077	56.686	63.937	1.128
controlled burn	83.164	88.173	1.060	52.956	57.702	1.090
elephants dream	83.214	87.599	1.053	54.001	58.475	1.083
moving zone plate	80.942	83.727	1.034	52.593	55.346	1.052
speed bag	74.539	76.544	1.027	48.129	50.138	1.042
tractor	84.062	86.571	1.030	54.589	57.542	1.054
Moyenne	85.324	89.302	1.046	57.907	61.909	1.070

La courbe d'accélération du nouvel algorithme de l'extraction de l'élément *total_zeros* en fonction du débit d'encodage pour la résolution 360p est illustrée à la figure I-1. Les données de cette figure proviennent des résultats obtenus pour les tableaux I-1 et I-2.

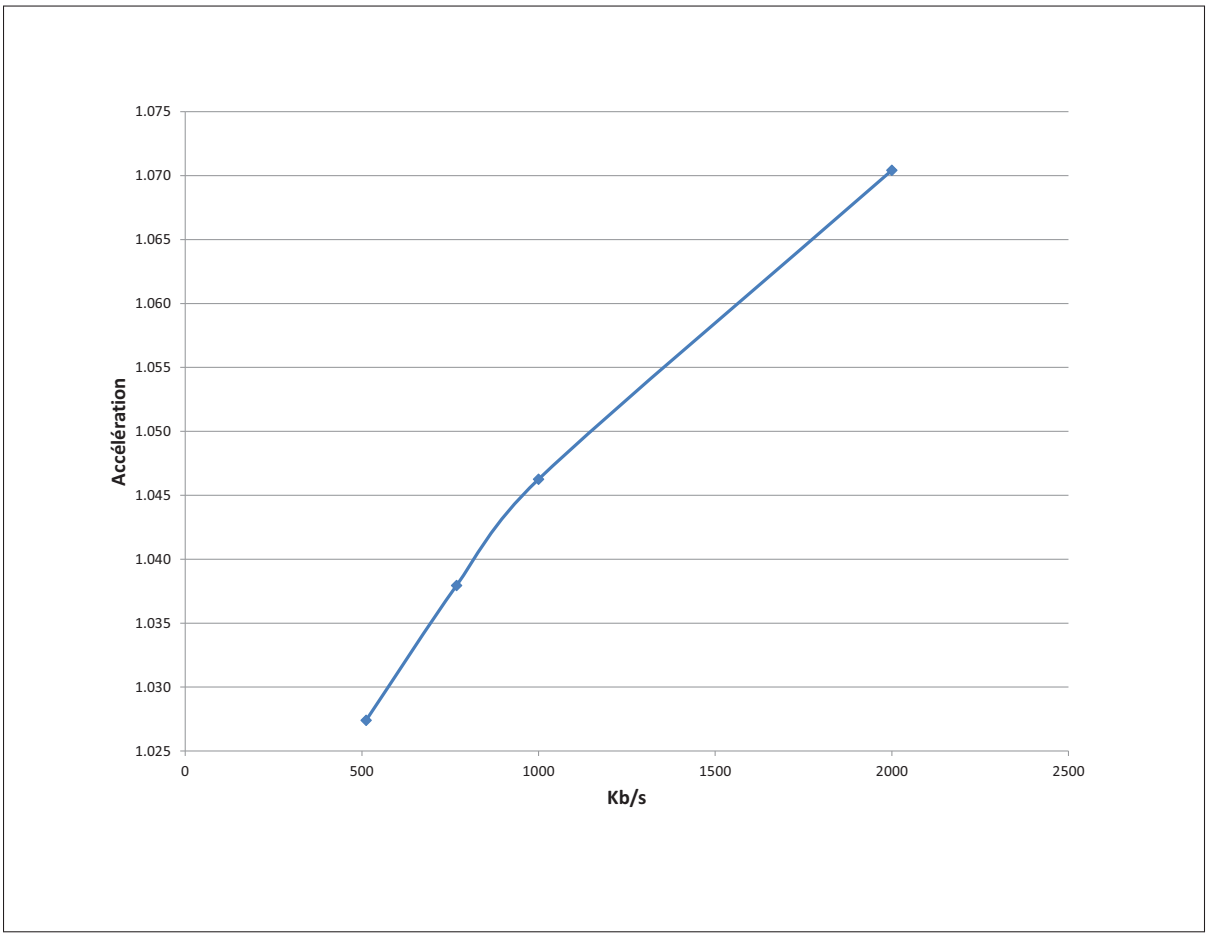


Figure-A I-1 Courbe d'accélération du nouvel algorithme pour l'extraction de l'élément *total_zeros* pour la résolution 360p en fonction du débit d'encodage.

1.2 Résolution NTSC

Les vitesses d'exécution moyennes en fps et les accélérations du processus de décodage de l'entropie de type CAVLC pour des séquences vidéo utilisant la résolution NTSC sont présentées dans les tableaux I-3 et I-4. Le tableau I-3 contient les résultats pour les séquences vidéo ayant un débit d'encodage de 512 kb/s et 768 kb/s. Le tableau I-4 contient les résultats pour les séquences vidéo ayant un débit d'encodage de 1000 kb/s et 2000 kb/s.

Tableau-A I-3 Vitesses d'exécution moyennes (fps) et accélération moyenne du processus de décodage de l'entropie et de l'analyse syntaxique pour la résolution NTSC utilisant des séquences vidéos à 512 kb/s et 768 kb/s.

Séquence	**512 kb/s**			**768 kb/s**		
	R (fps)	P (fps)	A	R (fps)	P (fps)	A
bad apple	109.263	111.849	1.024	93.902	96.903	1.032
big buck bunny	99.954	102.661	1.027	83.627	87.307	1.044
controlled burn	97.537	99.801	1.023	80.905	83.529	1.032
elephants dream	99.479	101.222	1.018	82.209	84.511	1.028
moving zone plate	97.720	98.971	1.013	79.361	80.824	1.018
speed bag	91.103	92.207	1.012	74.425	75.636	1.016
tractor	100.360	101.485	1.011	82.347	83.572	1.015
Moyenne	99.345	101.171	1.018	82.397	84.612	1.027

Tableau-A I-4 Vitesses d'exécution moyennes (fps) et accélération moyenne du processus de décodage de l'entropie et de l'analyse syntaxique pour la résolution NTSC utilisant des séquences vidéos à 1000 kb/s et 2000 kb/s.

Séquence	**1000 kb/s**			**2000 kb/s**		
	R (fps)	P (fps)	A	R (fps)	P (fps)	A
bad apple	83.919	87.083	1.038	64.271	67.226	1.046
big buck bunny	73.004	77.131	1.057	49.192	54.273	1.103
controlled burn	70.509	73.351	1.040	46.535	49.593	1.066
elephants dream	71.549	74.326	1.039	47.699	50.730	1.064
moving zone plate	68.147	69.695	1.023	44.643	46.440	1.040
speed bag	64.239	65.515	1.020	41.975	43.302	1.032
tractor	71.762	73.108	1.019	47.875	49.489	1.034
Moyenne	71.876	74.316	1.034	48.884	51.579	1.055

La courbe d'accélération du nouvel algorithme de l'extraction de l'élément *total_zeros* en fonction du débit d'encodage pour la résolution NTSC est illustrée à la figure I-2. Les données de cette figure proviennent des résultats obtenus pour les tableaux I-3 et I-4.

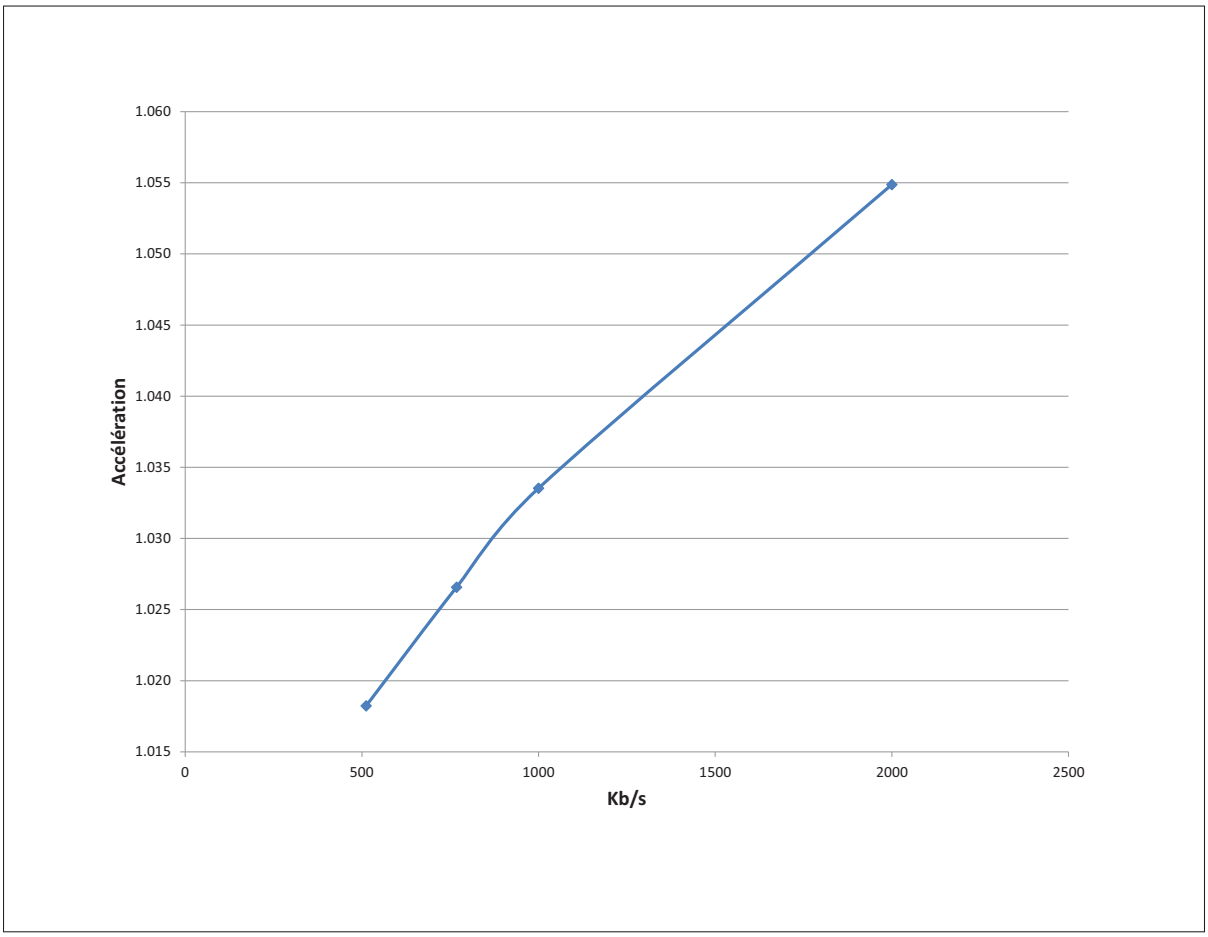


Figure-A I-2 Courbe d'accélération du nouvel algorithme pour l'extraction de l'élément *total_zeros* pour la résolution NTSC en fonction du débit d'encodage.

2 Intervalle de notification de la position d'un indice de MB

2.1 Résolution 360p

Les vitesses d'exécution moyennes en fps du processus de la reconstruction et du filtrage anti-bloc pour différentes tailles d'intervalle de notification en macrobloc pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 512 kb/s, 768 kb/s, 1000 kb/s, et 2000 kb/s sont présentés dans les tableaux I-5, I-6, I-7, et I-8 respectivement.

Tableau-A I-5 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution 360p utilisant des séquences vidéos à 512 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	88.402	88.133	86.596	84.840	84.137	79.713	67.465	56.348
big buck bunny	79.766	79.374	77.940	76.434	75.795	72.872	63.857	53.571
controlled burn	85.819	85.566	84.165	82.422	81.718	78.620	68.663	57.409
elephants dream	72.225	71.658	70.051	69.049	68.429	67.022	58.897	48.301
moving zone plate	72.023	71.457	69.861	68.828	68.467	66.557	58.586	48.137
speed bag	76.282	75.486	74.050	72.879	71.733	68.841	59.340	49.973
tractor	68.878	68.241	66.836	65.970	65.053	63.236	55.542	45.624
Moyenne	77.628	77.131	75.643	74.346	73.619	70.980	61.764	51.338

Tableau-A I-6 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution 360p utilisant des séquences vidéos à 768 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	87.697	87.213	85.837	84.050	83.225	79.048	67.078	56.252
big buck bunny	77.691	77.077	75.818	74.382	73.693	70.796	61.940	52.034
controlled burn	82.363	81.818	80.550	79.323	78.146	74.936	65.237	54.625
elephants dream	70.640	69.875	68.525	67.459	66.633	65.431	57.358	47.067
moving zone plate	69.648	68.599	67.406	66.530	65.871	64.289	56.773	46.718
speed bag	73.358	72.387	71.135	70.091	68.968	66.167	57.214	47.962
tractor	67.505	66.443	65.507	64.632	63.619	61.667	54.111	44.511
Moyenne	75.557	74.773	73.540	72.352	71.451	68.905	59.959	49.881

La courbe des vitesses d'exécution moyennes en fps du processus de la reconstruction et du filtrage antibloc en fonction de la taille de l'intervalle de notification en macrobloc utilisée pour

Tableau-A I-7 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution 360p utilisant des séquences vidéos à 1000 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	87.358	86.589	85.424	83.499	82.917	78.723	66.954	56.065
big buck bunny	76.460	75.768	74.533	73.096	72.412	69.407	60.645	51.040
controlled burn	79.856	79.189	77.940	76.338	75.684	72.397	62.928	52.679
elephants dream	69.397	68.538	67.171	66.168	65.181	64.118	56.095	46.084
moving zone plate	68.089	67.031	65.800	64.907	64.118	62.723	55.444	45.651
speed bag	71.189	70.178	68.942	67.925	66.821	64.258	55.645	46.608
tractor	66.291	65.233	64.219	63.341	62.343	60.437	52.879	43.558
Moyenne	74.091	73.218	72.004	70.753	69.925	67.438	58.656	48.812

Tableau-A I-8 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution 360p utilisant des séquences vidéos à 2000 kb/s.

Séquence	**1 MB**	**2 MB**	**4 MB**	**5 MB**	**6 MB**	**8 MB**	**12 MB**	**16 MB**
bad apple	87.188	86.306	85.310	83.373	82.800	78.573	67.085	56.293
big buck bunny	72.266	71.378	70.348	68.908	68.359	65.386	56.841	47.821
controlled burn	72.297	71.168	70.307	68.682	68.262	65.159	56.637	47.189
elephants dream	65.309	64.333	63.029	62.155	60.863	59.910	52.109	42.802
moving zone plate	63.773	62.749	61.493	60.577	59.498	58.130	51.393	42.291
speed bag	65.221	64.138	63.070	62.054	61.112	58.824	51.061	42.683
tractor	62.300	61.135	60.128	59.268	58.358	56.389	49.268	40.625
Moyenne	69.765	68.744	67.669	66.431	65.607	63.196	54.913	45.672

Tableau-A I-9 Sommaire des vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution 360p.

Sommaire	**1 MB**	**2 MB**	**4 MB**	**5 MB**	**6 MB**	**8 MB**	**12 MB**	**16 MB**
512 kb/s	77.628	77.131	75.643	74.346	73.619	70.980	61.764	51.338
768 kb/s	75.557	74.773	73.540	72.352	71.451	68.905	59.959	49.881
1000 kb/s	74.091	73.218	72.004	70.753	69.925	67.438	58.656	48.812
2000 kb/s	69.765	68.744	67.669	66.431	65.607	63.196	54.913	45.672
Moyenne	74.260	73.466	72.214	70.971	70.151	67.630	58.823	48.926

la synchronisation de tâches avec dépendances de données pour la résolution 360p est illustrée à la figure I-3. Les données de cette figure proviennent des résultats obtenus pour le tableau I-9.

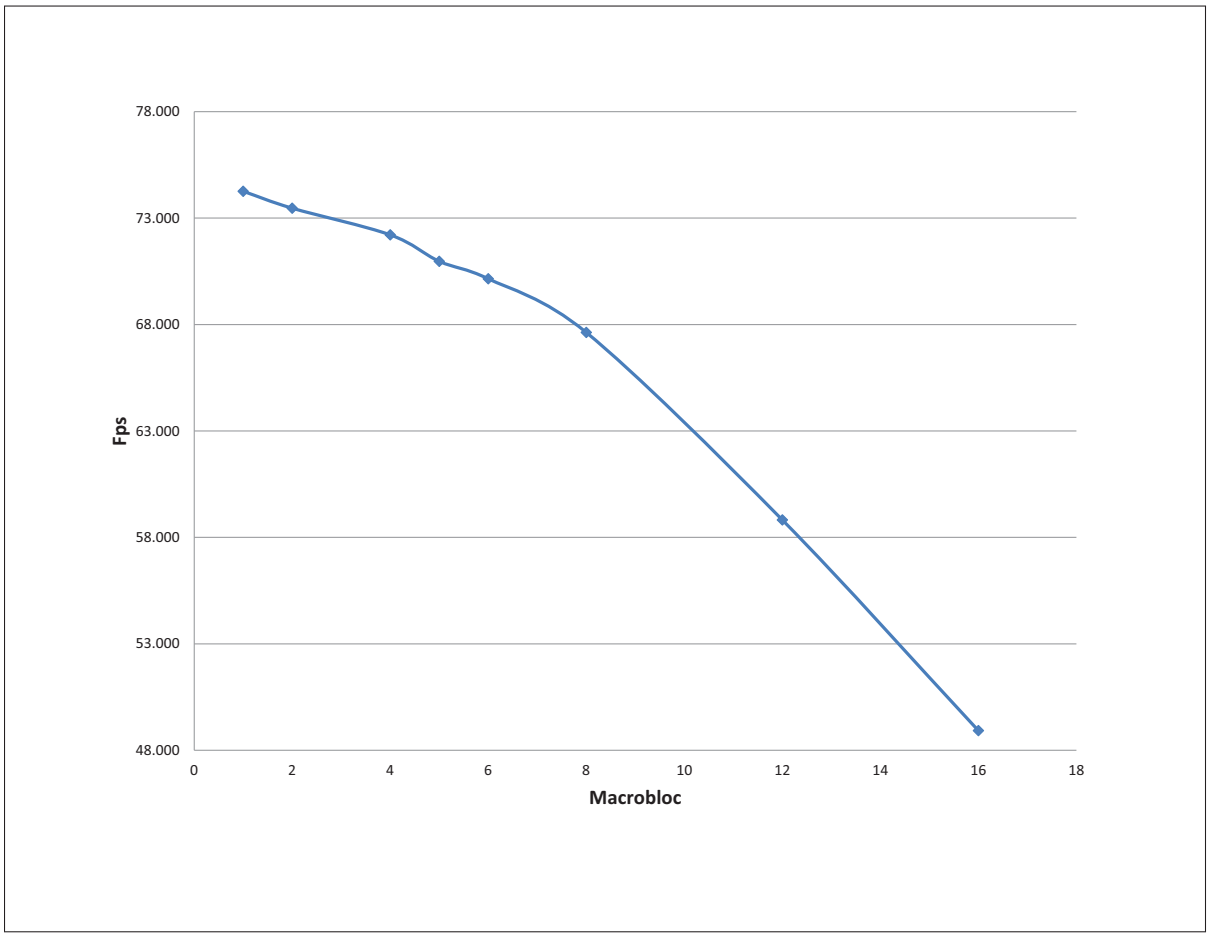


Figure-A I-3 Courbe des vitesses d'exécution moyennes du processus de la reconstruction et du filtrage antibloc en fonction de la taille de l'intervalle de notification en macrobloc pour la résolution 360p.

2.2 Résolution NTSC

Les vitesses d'exécution moyennes en fps du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 512 kb/s, 768 kb/s, 1000 kb/s, et 2000 kb/s sont présentés dans les tableaux I-10, I-11, I-12, et I-13 respectivement.

Tableau-A I-10 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution NTSC utilisant des séquences vidéos à 512 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	61.609	61.861	60.481	60.586	59.670	58.067	51.828	42.820
big buck bunny	57.154	57.335	56.012	56.069	55.393	54.230	50.786	41.880
controlled burn	61.244	61.588	60.231	60.310	59.495	58.157	54.400	45.125
elephants dream	50.857	50.829	49.480	49.535	48.512	47.721	46.048	37.430
moving zone plate	51.547	51.520	50.071	50.101	49.397	48.788	46.177	37.292
speed bag	53.990	54.023	52.794	52.844	52.129	50.708	46.972	39.209
tractor	47.573	47.449	46.319	46.385	45.474	44.704	42.886	34.988
Moyenne	54.853	54.944	53.627	53.690	52.867	51.768	48.442	39.821

Tableau-A I-11 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution NTSC utilisant des séquences vidéos à 768 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	61.132	61.396	60.018	60.143	59.266	57.734	51.898	42.786
big buck bunny	55.318	55.513	54.320	54.346	53.767	52.540	49.336	40.755
controlled burn	58.802	59.089	57.785	57.801	56.991	55.647	51.959	43.043
elephants dream	49.858	49.852	48.629	48.676	47.669	46.823	45.197	36.747
moving zone plate	49.783	49.801	48.594	48.659	47.934	47.273	45.349	36.718
speed bag	51.985	51.981	50.837	50.806	50.188	48.881	45.446	37.826
tractor	46.312	46.267	45.339	45.369	44.566	43.867	42.147	34.408
Moyenne	53.313	53.414	52.217	52.257	51.483	50.395	47.333	38.898

La courbe des vitesses d'exécution moyennes en fps du processus de la reconstruction et du filtrage antibloc en fonction de la taille de l'intervalle de notification en macrobloc utilisée

Tableau-A I-12 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution NTSC utilisant des séquences vidéos à 1000 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	60.732	60.955	59.786	59.746	59.000	57.415	51.832	42.750
big buck bunny	54.281	54.448	53.258	53.371	52.772	51.542	48.409	40.008
controlled burn	57.058	57.303	56.045	56.095	55.304	53.981	50.292	41.635
elephants dream	49.209	49.240	48.025	48.107	47.052	46.131	44.540	36.227
moving zone plate	48.728	48.746	47.580	47.624	46.842	45.981	44.549	36.142
speed bag	50.613	50.601	49.455	49.456	48.815	47.539	44.375	36.865
tractor	45.675	45.611	44.693	44.691	43.928	43.180	41.441	33.877
Moyenne	52.328	52.415	51.263	51.299	50.530	49.396	46.491	38.215

Tableau-A I-13 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution NTSC utilisant des séquences vidéos à 2000 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	60.457	60.661	59.460	59.574	58.665	57.113	51.853	42.721
big buck bunny	51.642	51.748	50.563	50.612	50.065	48.826	45.838	37.881
controlled burn	52.004	52.151	50.914	50.954	50.303	49.118	45.708	37.696
elephants dream	47.082	47.051	45.851	45.856	45.138	43.827	42.096	34.221
moving zone plate	45.986	45.875	44.698	44.721	44.198	43.116	41.933	34.145
speed bag	46.566	46.503	45.393	45.410	44.865	43.684	40.984	33.965
tractor	43.518	43.413	42.368	42.408	41.781	40.874	39.139	31.992
Moyenne	49.608	49.629	48.464	48.505	47.859	46.651	43.936	36.089

Tableau-A I-14 Sommaire des vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour la résolution NTSC.

Sommaire	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
512 kb/s	54.853	54.944	53.627	53.690	52.867	51.768	48.442	39.821
768 kb/s	53.313	53.414	52.217	52.257	51.483	50.395	47.333	38.898
1000 kb/s	52.328	52.415	51.263	51.299	50.530	49.396	46.491	38.215
2000 kb/s	49.608	49.629	48.464	48.505	47.859	46.651	43.936	36.089
Moyenne	52.526	52.600	51.393	51.438	50.685	49.552	46.551	38.255

pour la synchronisation de tâches avec dépendances de données pour la résolution NTSC est illustrée à la figure I-4. Les données de cette figure proviennent des résultats obtenus pour le tableau I-14.

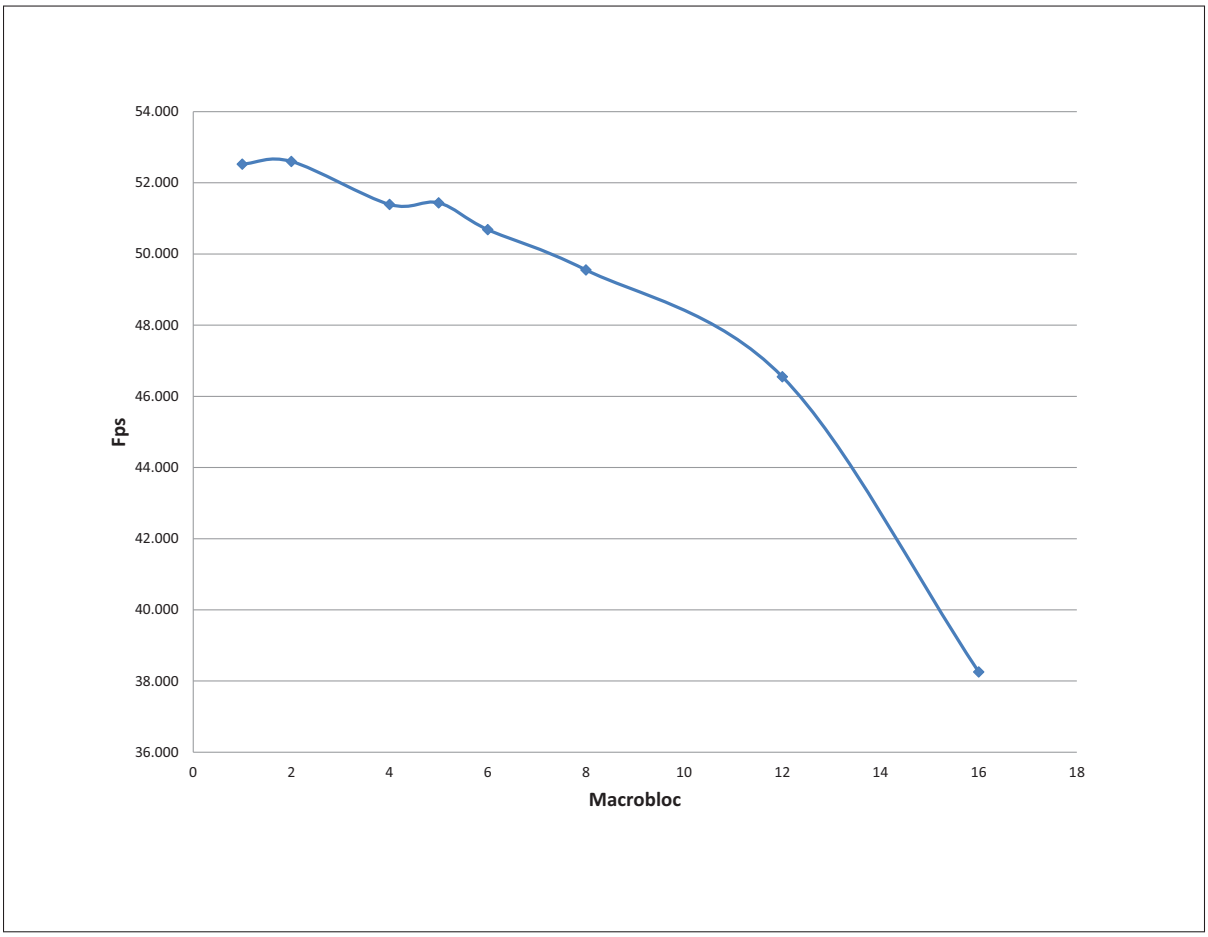


Figure-A I-4 Courbe des vitesses d'exécution moyennes du processus de la reconstruction et du filtrage antibloc en fonction de la taille de l'intervalle de notification en macrobloc pour la résolution NTSC.

2.3 Résolution 720p

Les vitesses d'exécution moyennes en fps du processus de la reconstruction et du filtrage anti-bloc pour différentes tailles d'intervalle de notification en macrobloc pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1000 kb/s, 1500 kb/s, 2000 kb/s, et 4000 kb/s sont présentés dans les tableaux I-15, I-16, I-17, et I-18 respectivement.

Tableau-A I-15 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1000 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	41.646	42.472	42.142	41.615	41.061	38.205	30.966	25.566
big buck bunny	38.682	39.274	38.859	38.405	38.148	36.805	30.695	25.232
controlled burn	40.894	41.647	41.306	40.905	40.640	39.228	32.689	27.205
elephants dream	35.334	35.674	35.126	34.700	34.438	33.581	28.548	23.316
moving zone plate	35.247	35.657	35.159	34.719	34.463	33.611	28.492	23.253
speed bag	36.519	36.947	36.491	36.148	35.710	33.905	27.740	22.852
tractor	31.683	31.867	31.343	30.976	30.708	29.880	24.924	20.246
Moyenne	37.144	37.648	37.204	36.781	36.453	35.031	29.151	23.953

Tableau-A I-16 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1500 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	41.407	42.161	41.827	41.293	40.786	38.206	31.145	25.739
big buck bunny	37.742	38.254	37.814	37.389	37.155	35.832	29.804	24.459
controlled burn	39.561	40.200	39.825	39.419	39.156	37.689	31.305	25.983
elephants dream	34.994	35.309	34.773	34.331	34.084	33.222	28.255	23.084
moving zone plate	34.913	35.281	34.780	34.367	34.117	33.226	28.165	23.008
speed bag	35.327	35.669	35.224	34.864	34.511	32.835	26.910	22.193
tractor	31.301	31.465	30.956	30.587	30.355	29.480	24.645	20.029
Moyenne	36.464	36.906	36.457	36.036	35.738	34.356	28.604	23.499

Tableau-A I-17 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 2000 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	41.488	42.266	41.917	41.340	40.830	38.256	31.286	25.901
big buck bunny	37.195	37.678	37.250	36.837	36.577	35.223	29.250	23.956
controlled burn	38.566	39.131	38.760	38.349	38.077	36.606	30.298	25.115
elephants dream	34.782	35.077	34.569	34.127	33.860	33.007	28.005	22.897
moving zone plate	34.388	34.696	34.201	33.800	33.558	32.696	27.779	22.708
speed bag	34.317	34.643	34.184	33.836	33.476	31.890	26.172	21.596
tractor	30.976	31.135	30.639	30.297	30.036	29.163	24.400	19.831
Moyenne	35.959	36.375	35.931	35.512	35.202	33.834	28.170	23.143

Tableau-A I-18 Vitesses d'exécution moyennes (fps) du processus de la reconstruction et du filtrage antibloc pour différentes tailles d'intervalle de notification en macrobloc pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 4000 kb/s.

Séquence	1 MB	2 MB	4 MB	5 MB	6 MB	8 MB	12 MB	16 MB
bad apple	41.455	42.196	41.848	41.195	40.674	38.342	31.494	26.140
big buck bunny	35.733	36.145	35.691	35.293	35.029	33.653	27.761	22.668
controlled burn	35.921	36.328	35.912	35.545	35.247	33.791	27.843	22.980
elephants dream	33.521	33.794	33.253	32.840	32.584	31.753	26.632	21.773
moving zone plate	33.084	33.319	32.841	32.433	32.204	31.321	26.643	21.776
speed bag	31.730	31.952	31.517	31.179	30.866	29.551	24.206	19.945
tractor	29.793	29.927	29.453	29.126	28.856	28.003	23.387	19.014
Moyenne	34.462	34.809	34.359	33.944	33.637	32.345	26.852	22.042

3 Répartition du nombre de coeurs DSP alloués pour chaque processus

3.1 Résolution 360p

Dans cette sous-section, nous présentons les tableaux pour le sommaire des vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant différents débits d'encodage pour la résolution 360p et où le décodeur utilise de 1 à 9 coeurs DSP. À l'intérieur de ces tableaux, le paramètre f_1 représente le nombre de trames par seconde que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique, et le paramètre f_2 représente le nombre de trames par seconde que peut traiter le processus de la reconstruction et du filtrage antibloc. Chacun des processus f_1 et f_2 utilise le nombre de coeurs indiqué dans la colonne des tables. Notons aussi que pour le processus f_1, nous n'avons pas indiqué les vitesses pour des nombres élevés de coeurs puisque ce processus, disposant de quelques coeurs, est déjà plus rapide que le processus f_2 même lorsque ce dernier dispose de 9 coeurs.

Tableau-A I-19 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 512 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	144.124	30.318	189.871	49.190	200.939	69.751
big buck bunny	129.568	26.923	158.406	44.292	162.036	62.738
controlled burn	124.671	29.231	155.321	47.803	166.602	67.761
elephants dream	124.109	23.418	145.856	39.187	149.316	56.088
moving zone plate	120.028	23.425	141.151	39.120	141.402	56.313
speed bag	111.983	25.189	147.442	41.749	160.215	59.334
tractor	122.841	21.885	154.992	37.170	155.503	53.344
Moyenne	125.332	25.770	156.148	42.644	162.288	60.761

Tableau-A I-20 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 512 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	209.998	88.133	211.576	99.458	-	111.528
big buck bunny	164.086	79.374	164.131	90.194	-	101.905
controlled burn	172.928	85.566	174.644	96.970	-	109.000
elephants dream	151.006	71.658	152.414	81.802	-	93.318
moving zone plate	142.545	71.457	142.540	81.362	-	92.894
speed bag	165.631	75.486	168.046	86.294	-	97.202
tractor	156.756	68.241	156.776	78.095	-	89.242
Moyenne	166.136	77.131	167.161	87.739	-	99.298

Tableau-A I-21 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 512 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	119.638	-	131.827	-	128.249
big buck bunny	-	109.244	-	121.111	-	119.678
controlled burn	-	117.305	-	127.954	-	127.104
elephants dream	-	100.002	-	112.671	-	112.824
moving zone plate	-	98.976	-	112.060	-	112.480
speed bag	-	105.070	-	115.385	-	113.488
tractor	-	96.879	-	107.961	-	108.632
Moyenne	-	106.731	-	118.424	-	117.494

Tableau-A I-22 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 768 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	122.059	30.049	176.805	48.763	209.336	68.974
big buck bunny	107.297	26.112	152.621	43.068	170.451	60.928
controlled burn	101.841	27.718	143.322	45.679	157.206	64.710
elephants dream	101.167	22.905	148.483	38.341	158.104	54.757
moving zone plate	97.233	22.523	156.250	37.673	165.430	53.734
speed bag	89.485	24.054	132.565	40.050	152.833	56.861
tractor	99.563	21.405	160.422	36.365	167.573	51.870
Moyenne	102.664	24.967	152.924	41.420	168.705	58.833

Tableau-A I-23 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 768 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	227.757	87.213	232.096	98.754	-	110.080
big buck bunny	175.345	77.077	176.274	87.773	-	99.172
controlled burn	168.576	81.818	171.159	92.884	-	104.549
elephants dream	161.087	69.875	163.037	79.980	-	91.105
moving zone plate	166.793	68.599	167.168	78.371	-	89.227
speed bag	166.763	72.387	170.379	82.901	-	93.575
tractor	169.415	66.443	171.256	76.514	-	87.179
Moyenne	176.534	74.773	178.767	85.311	-	96.412

Tableau-A I-24 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 768 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	118.208	-	130.284	-	126.859
big buck bunny	-	106.591	-	118.350	-	116.793
controlled burn	-	112.618	-	123.180	-	122.030
elephants dream	-	97.646	-	110.009	-	110.141
moving zone plate	-	95.316	-	107.767	-	108.053
speed bag	-	101.327	-	111.305	-	109.682
tractor	-	95.167	-	105.685	-	106.450
Moyenne	-	103.839	-	115.226	-	114.287

Tableau-A I-25 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 1000 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	108.884	29.900	167.897	48.553	199.693	68.354
big buck bunny	93.618	25.587	138.222	42.260	167.863	59.847
controlled burn	88.173	26.674	134.468	44.134	157.870	62.521
elephants dream	87.599	22.485	133.208	37.693	161.147	53.678
moving zone plate	83.727	21.940	134.253	36.768	161.922	52.308
speed bag	76.544	23.208	122.101	38.771	149.141	55.097
tractor	86.571	20.934	130.606	35.595	176.281	50.904
Moyenne	89.302	24.390	137.251	40.539	167.702	57.530

Tableau-A I-26 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 1000 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	227.795	86.589	245.884	96.751	-	109.435
big buck bunny	180.947	75.768	183.035	86.216	-	97.719
controlled burn	169.413	79.189	176.149	90.114	-	101.662
elephants dream	166.254	68.538	169.372	78.388	-	89.489
moving zone plate	163.862	67.031	164.294	76.317	-	86.978
speed bag	165.395	70.178	177.142	80.480	-	91.167
tractor	181.878	65.233	184.382	75.071	-	85.693
Moyenne	179.363	73.218	185.751	83.334	-	94.592

Tableau-A I-27 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 1000 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	117.770	-	129.609	-	126.402
big buck bunny	-	105.010	-	116.371	-	114.928
controlled burn	-	109.518	-	119.647	-	118.837
elephants dream	-	95.911	-	108.073	-	108.205
moving zone plate	-	93.110	-	104.911	-	105.111
speed bag	-	98.697	-	108.449	-	107.265
tractor	-	93.495	-	103.735	-	104.494
Moyenne	-	101.930	-	112.971	-	112.177

Tableau-A I-28 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 2000 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	90.223	29.904	146.275	48.581	188.952	68.217
big buck bunny	63.937	23.847	107.035	39.631	136.932	56.233
controlled burn	57.702	23.569	98.940	39.604	131.913	55.847
elephants dream	58.475	21.018	100.733	35.406	131.271	50.287
moving zone plate	55.346	20.234	96.282	34.149	137.760	48.410
speed bag	50.138	20.918	87.878	35.249	118.576	50.171
tractor	57.542	19.484	99.125	33.272	137.533	47.559
Moyenne	61.909	22.711	105.181	37.985	162.288	53.818

Tableau-A I-29 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 2000 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	214.256	86.306	231.803	96.447	-	108.921
big buck bunny	157.160	71.378	165.135	81.537	-	92.577
controlled burn	153.984	71.168	174.874	81.333	-	92.436
elephants dream	156.911	64.333	176.776	73.745	-	84.247
moving zone plate	159.243	62.749	178.251	70.774	-	80.607
speed bag	143.283	64.138	160.699	73.584	-	83.860
tractor	150.195	61.135	175.533	70.517	-	80.533
Moyenne	162.147	68.744	180.439	78.277	-	89.026

Tableau-A I-30 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p et un débit d'encodage de 2000 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	117.199	-	128.981	-	125.655
big buck bunny	-	99.973	-	110.696	-	109.596
controlled burn	-	100.024	-	109.645	-	109.204
elephants dream	-	90.639	-	101.910	-	102.242
moving zone plate	-	86.478	-	98.669	-	98.445
speed bag	-	91.262	-	100.706	-	100.229
tractor	-	88.167	-	97.718	-	98.438
Moyenne	-	96.249	-	106.904	-	106.258

Tableau-A I-31 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	125.332	25.770	156.148	42.644	162.288	60.761
1500 kb/s	102.664	24.967	152.924	41.420	168.705	58.833
2000 kb/s	89.302	24.390	137.251	40.539	167.702	57.530
4000 kb/s	61.909	22.711	105.181	37.985	162.288	53.818
minimum	61.909	22.711	105.181	37.985	162.288	53.818

Tableau-A I-32 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	166.136	77.131	167.161	87.739	-	99.298
1500 kb/s	176.534	74.773	178.767	85.311	-	96.412
2000 kb/s	179.363	73.218	185.751	83.334	-	94.592
4000 kb/s	162.147	68.744	180.439	78.277	-	89.026
minimum	162.147	68.744	167.161	78.277	-	89.026

Tableau-A I-33 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 360p où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	-	106.731	-	118.424	-	117.494
1500 kb/s	-	103.839	-	115.226	-	114.287
2000 kb/s	-	101.930	-	112.971	-	112.177
4000 kb/s	-	96.249	-	106.904	-	106.258
minimum	-	96.249	-	106.904	-	106.258

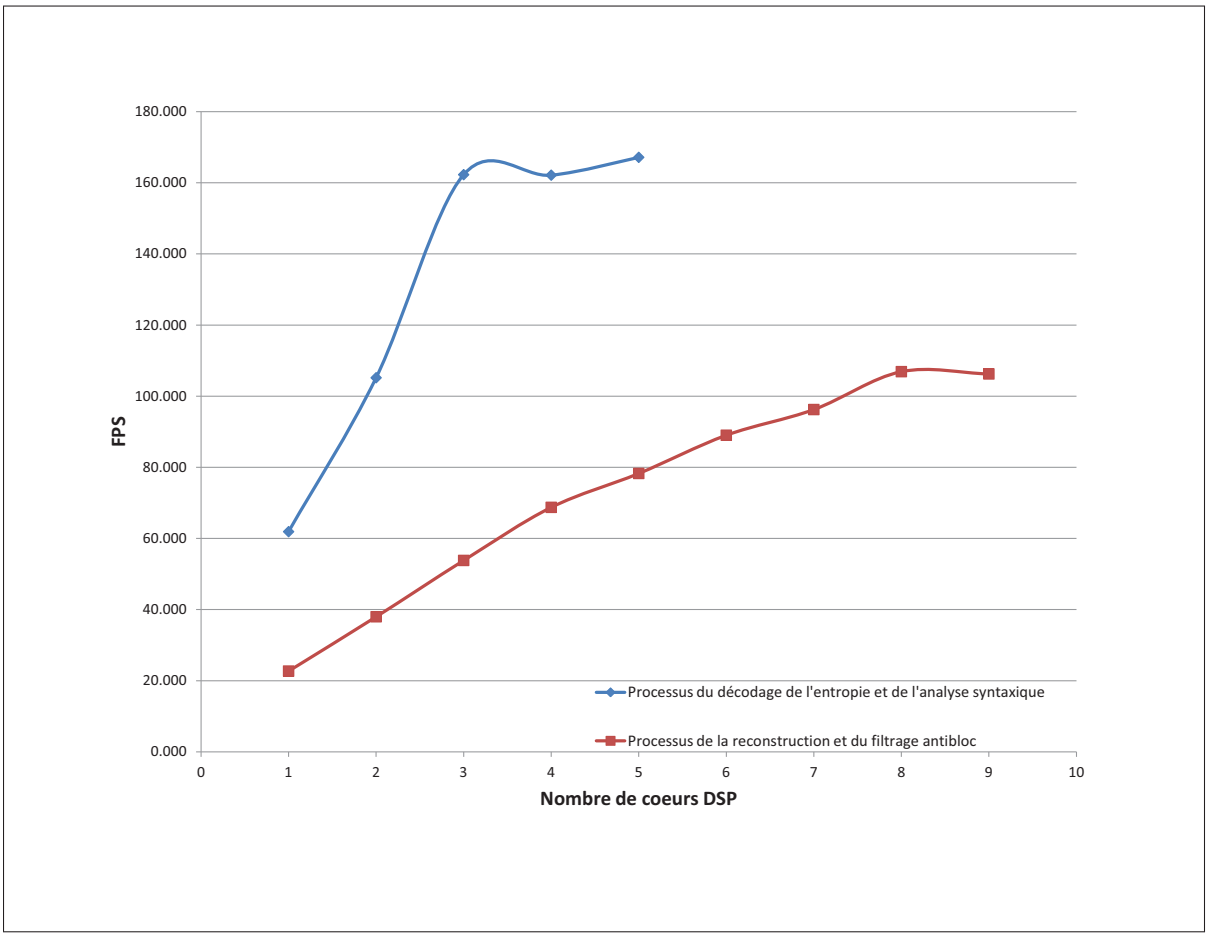


Figure-A I-5 Courbes des vitesses d'exécution moyennes pour la résolution 360p utilisant différents nombres de coeurs DSP pour le processus du décodage de l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc.

Les courbes des vitesses d'exécution moyennes pour la résolution 360p utilisant différents nombres de coeurs DSP pour le processus du décodage de l'entropie et de l'analyse syntaxique et pour le processus de la reconstruction et du filtrage antibloc sont présentées à la figure I-5. Les données de cette figure proviennent des résultats obtenus pour les tableaux I-31, I-32 et I-33.

3.2 Résolution NTSC

Dans cette sous-section, nous présentons les tableaux pour le sommaire des vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant différents débits d'encodage pour la résolution NTSC et où le décodeur utilise de 1 à 9 coeurs DSP. À l'intérieur de ces tableaux,

Tableau-A I-34 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 512 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	111.849	20.698	147.218	35.016	154.809	49.924
big buck bunny	102.661	18.813	126.718	32.175	130.203	46.145
controlled burn	99.801	20.651	123.531	34.915	132.309	49.813
elephants dream	101.222	16.117	121.688	28.092	124.106	40.706
moving zone plate	98.971	16.333	118.365	28.432	118.241	41.198
speed bag	92.207	17.641	122.104	30.332	132.401	43.467
tractor	101.485	14.901	127.526	26.094	127.850	37.830
Moyenne	101.171	17.879	126.736	30.722	131.417	44.155

Tableau-A I-35 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 512 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	159.814	61.861	161.152	72.956	-	81.883
big buck bunny	132.452	57.335	132.632	67.950	-	76.694
controlled burn	137.417	61.588	138.979	72.657	-	81.547
elephants dream	126.484	50.829	127.033	60.858	-	69.248
moving zone plate	119.543	51.520	119.642	61.518	-	70.037
speed bag	136.974	54.023	138.886	64.112	-	72.273
tractor	129.230	47.449	129.277	56.928	-	64.867
Moyenne	134.559	54.944	135.372	65.283	-	73.793

le paramètre f_1 représente le nombre de trames par seconde que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique, et le paramètre f_2 représente le nombre de trames par seconde que peut traiter le processus de la reconstruction et du filtrage antibloc. Chacun des processus f_1 et f_2 utilise le nombre de coeurs indiqué dans la colonne des tables. Notons aussi que pour le processus f_1, nous n'avons pas indiqué les vitesses pour des nombres élevés de coeurs puisque ce processus, disposant de quelques coeurs, est déjà plus rapide que le processus f_2 même lorsque ce dernier dispose de 9 coeurs.

Tableau-A I-36 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 512 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	88.259	-	96.726	-	97.047
big buck bunny	-	82.577	-	91.572	-	92.705
controlled burn	-	87.692	-	96.464	-	97.585
elephants dream	-	74.387	-	83.566	-	85.271
moving zone plate	-	75.228	-	83.850	-	85.531
speed bag	-	78.415	-	86.216	-	87.286
tractor	-	69.836	-	78.484	-	80.613
Moyenne	-	79.485	-	88.125	-	89.434

Tableau-A I-37 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 768 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	96.903	20.647	143.858	34.930	166.963	49.700
big buck bunny	87.307	18.220	125.096	31.265	138.822	44.844
controlled burn	83.529	19.654	116.007	33.425	127.199	47.685
elephants dream	84.511	15.895	122.991	27.715	133.337	40.102
moving zone plate	80.824	15.961	131.044	27.792	137.168	40.228
speed bag	75.636	16.950	112.806	29.244	131.052	41.944
tractor	83.572	14.687	136.462	25.765	143.157	37.289
Moyenne	84.612	17.431	126.895	30.019	139.671	43.113

Tableau-A I-38 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 768 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	176.524	61.396	181.559	72.497	-	81.289
big buck bunny	143.824	55.513	144.786	65.965	-	74.609
controlled burn	135.409	59.089	138.110	69.717	-	78.353
elephants dream	136.588	49.852	137.963	59.951	-	68.185
moving zone plate	139.084	49.801	139.574	60.052	-	68.105
speed bag	141.630	51.981	145.044	61.866	-	69.839
tractor	145.415	46.267	146.111	56.071	-	63.792
Moyenne	145.496	53.414	147.592	63.731	-	72.025

Tableau-A I-39 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 768 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	87.619	-	95.970	-	96.464
big buck bunny	-	80.539	-	89.127	-	90.375
controlled burn	-	84.544	-	92.947	-	94.136
elephants dream	-	73.370	-	82.261	-	83.942
moving zone plate	-	73.069	-	81.619	-	83.189
speed bag	-	75.942	-	83.603	-	84.606
tractor	-	68.957	-	77.191	-	79.556
Moyenne	-	77.720	-	86.103	-	87.467

Tableau-A I-40 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 1000 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	87.083	20.547	136.471	34.735	162.613	49.382
big buck bunny	77.131	17.829	114.406	30.662	136.929	43.990
controlled burn	73.351	19.003	111.798	32.448	129.869	46.295
elephants dream	74.326	15.705	111.736	27.423	136.232	39.635
moving zone plate	69.695	15.635	108.696	27.253	142.347	39.408
speed bag	65.515	16.454	104.726	28.437	128.119	40.801
tractor	73.108	14.463	113.450	25.397	152.881	36.764
Moyenne	74.316	17.091	114.469	29.479	141.284	42.325

Tableau-A I-41 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 1000 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	182.140	60.955	193.071	72.097	-	80.977
big buck bunny	146.176	54.448	148.423	64.737	-	73.323
controlled burn	138.891	57.303	145.264	67.840	-	76.330
elephants dream	140.948	49.240	142.544	59.297	-	67.460
moving zone plate	143.983	48.746	144.760	58.666	-	66.551
speed bag	141.236	50.601	150.032	60.225	-	68.290
tractor	156.290	45.611	156.602	55.308	-	62.995
Moyenne	149.952	52.415	154.385	62.596	-	70.847

Tableau-A I-42 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 1000 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	87.369	-	95.534	-	96.169
big buck bunny	-	79.072	-	87.757	-	88.996
controlled burn	-	82.360	-	90.657	-	91.959
elephants dream	-	72.394	-	81.235	-	83.090
moving zone plate	-	71.046	-	79.925	-	81.637
speed bag	-	74.126	-	81.757	-	82.856
tractor	-	68.183	-	76.064	-	78.593
Moyenne	-	76.364	-	84.704	-	86.186

Tableau-A I-43 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 2000 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	67.226	20.467	113.192	34.572	146.374	49.167
big buck bunny	54.273	16.830	91.472	29.110	115.995	41.799
controlled burn	49.593	17.062	85.460	29.424	112.039	42.115
elephants dream	50.730	15.012	87.129	26.289	113.702	37.933
moving zone plate	46.440	14.609	80.242	25.527	112.785	36.894
speed bag	43.302	14.931	76.481	26.041	103.821	37.429
tractor	49.489	13.695	86.409	24.140	119.729	34.937
Moyenne	51.579	16.087	88.626	27.872	117.778	40.039

Tableau-A I-44 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 2000 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	171.177	60.661	185.997	71.452	-	80.249
big buck bunny	132.532	51.748	140.538	61.617	-	69.963
controlled burn	132.418	52.151	147.424	62.015	-	70.192
elephants dream	130.109	47.051	147.371	56.834	-	64.613
moving zone plate	124.592	45.875	151.108	54.403	-	61.943
speed bag	124.314	46.503	139.821	55.438	-	63.134
tractor	133.072	43.413	154.876	52.573	-	59.837
Moyenne	135.459	49.629	152.448	59.190	-	67.133

Tableau-A I-45 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC et un débit d'encodage de 2000 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	86.614	-	94.650	-	95.258
big buck bunny	-	75.744	-	83.997	-	85.188
controlled burn	-	75.904	-	83.785	-	85.150
elephants dream	-	69.543	-	77.757	-	79.766
moving zone plate	-	65.977	-	75.219	-	76.539
speed bag	-	68.669	-	75.891	-	77.398
tractor	-	65.186	-	72.417	-	74.891
Moyenne	-	72.520	-	80.531	-	82.027

Tableau-A I-46 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	101.171	17.879	126.736	30.722	131.417	44.155
1500 kb/s	84.612	17.431	126.895	30.019	139.671	43.113
2000 kb/s	74.316	17.091	114.469	29.479	141.284	42.325
4000 kb/s	51.579	16.087	88.626	27.872	117.778	40.039
minimum	51.579	16.087	88.626	27.872	117.778	40.039

Tableau-A I-47 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	134.559	54.944	135.372	65.283	-	73.793
1500 kb/s	145.496	53.414	147.592	63.731	-	72.025
2000 kb/s	149.952	52.415	154.385	62.596	-	70.847
4000 kb/s	135.459	49.629	152.448	59.190	-	67.133
minimum	134.559	49.629	135.372	59.190	-	67.133

Tableau-A I-48 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution NTSC où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
1000 kb/s	-	79.485	-	88.125	-	89.434
1500 kb/s	-	77.720	-	86.103	-	87.467
2000 kb/s	-	76.364	-	84.704	-	86.186
4000 kb/s	-	72.520	-	80.531	-	82.027
minimum	-	72.520	-	80.531	-	82.027

Les courbes des vitesses d'exécution moyennes pour la résolution NTSC utilisant différents nombres de coeurs DSP pour le processus du décodage de l'entropie et de l'analyse syntaxique et pour le processus de la reconstruction et du filtrage antibloc sont présentées à la figure I-6. Les données de cette figure proviennent des résultats obtenus pour les tableaux I-46, I-47 et I-48.

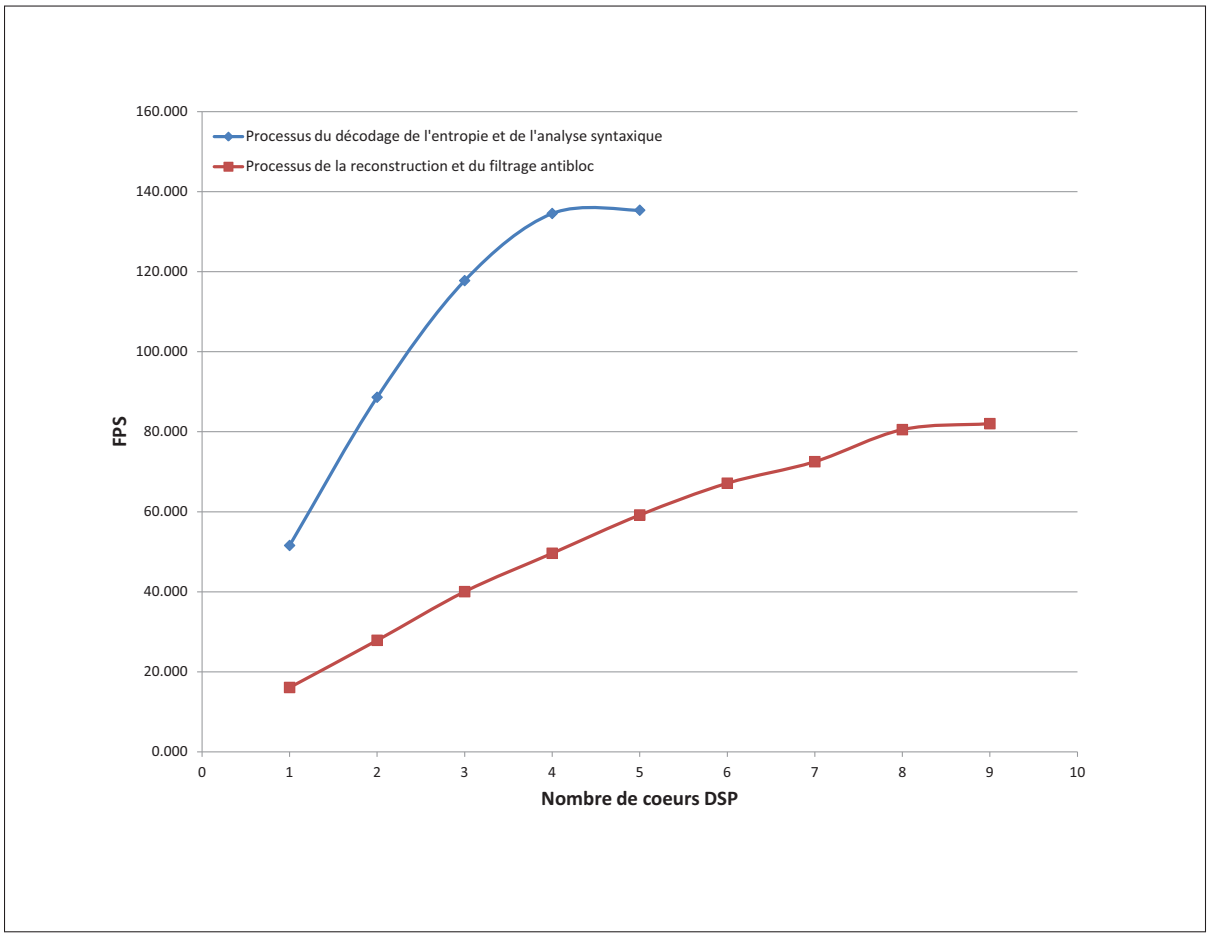


Figure-A I-6 Courbes des vitesses d'exécution moyennes pour la résolution NTSC utilisant différents nombres de coeurs DSP pour le processus du décodage de l'entropie et de l'analyse syntaxique et le processus de la reconstruction et du filtrage antibloc.

3.3 Résolution 720p

Dans cette sous-section, nous présentons les tableaux pour le sommaire des vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant différents débits d'encodage pour la résolution 720p et où le décodeur utilise de 1 à 9 coeurs DSP. À l'intérieur de ces tableaux, le paramètre f_1 représente le nombre de trames par seconde que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique, et le paramètre f_2 représente le nombre de trames par seconde que peut traiter le processus de la reconstruction et du filtrage antibloc. Chacun des processus f_1 et f_2 utilise le nombre de coeurs indiqué dans la colonne des tables. Notons aussi que pour le processus f_1, nous n'avons pas indiqué les vitesses pour des nombres

Tableau-A I-49 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1000 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	46.528	7.946	73.537	14.035	86.728	20.488
big buck bunny	43.013	7.045	64.569	12.600	73.748	18.490
controlled burn	41.703	7.748	61.403	13.722	69.829	20.065
elephants dream	43.506	6.032	66.461	10.936	81.246	16.176
moving zone plate	42.621	6.054	67.052	10.984	87.972	16.207
speed bag	39.517	6.593	63.341	11.817	75.173	17.372
tractor	44.302	5.338	73.444	9.720	89.785	14.401
Moyenne	43.027	6.679	67.115	11.973	80.640	17.600

Tableau-A I-50 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1000 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	95.115	24.894	97.267	31.436	-	34.974
big buck bunny	78.037	22.902	79.444	28.691	-	31.924
controlled burn	74.431	24.617	77.173	30.883	-	34.240
elephants dream	84.670	20.155	85.457	25.447	-	28.440
moving zone plate	90.035	20.383	90.305	25.445	-	28.522
speed bag	82.407	21.473	86.641	26.898	-	30.068
tractor	92.651	17.991	93.017	22.652	-	25.401
Moyenne	85.335	21.774	87.043	27.350	-	30.510

élevés de coeurs puisque ce processus, disposant de quelques coeurs, est déjà plus rapide que le processus f_2 même lorsque ce dernier dispose de 9 coeurs.

Tableau-A I-51 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1000 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	38.542	-	42.430	-	44.654
big buck bunny	-	35.347	-	39.285	-	41.959
controlled burn	-	37.771	-	41.647	-	44.120
elephants dream	-	31.792	-	35.662	-	38.560
moving zone plate	-	31.681	-	35.693	-	38.525
speed bag	-	33.305	-	36.950	-	39.357
tractor	-	28.407	-	31.879	-	34.581
Moyenne	-	33.835	-	37.649	-	40.251

Tableau-A I-52 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1500 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	41.154	7.940	68.325	14.022	85.379	20.415
big buck bunny	37.077	6.829	60.341	12.232	72.362	17.968
controlled burn	35.494	7.377	58.595	13.120	70.441	19.198
elephants dream	37.168	5.994	62.551	10.857	73.661	16.053
moving zone plate	35.565	6.003	62.241	10.879	75.928	16.079
speed bag	33.298	6.321	57.755	11.359	73.427	16.713
tractor	37.199	5.281	65.018	9.606	81.947	14.227
Moyenne	36.708	6.535	62.118	11.725	76.164	17.236

Tableau-A I-53 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1500 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	97.324	25.036	105.652	31.219	-	34.714
big buck bunny	82.019	22.288	86.494	27.880	-	31.088
controlled burn	80.171	23.651	84.265	29.581	-	32.884
elephants dream	86.254	19.977	90.502	25.242	-	28.212
moving zone plate	102.522	20.137	104.144	25.171	-	28.169
speed bag	84.252	20.728	91.629	25.869	-	28.999
tractor	102.730	17.737	107.740	22.369	-	25.091
Moyenne	90.753	21.365	95.775	26.762	-	29.880

Tableau-A I-54 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 1500 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	38.257	-	42.161	-	44.297
big buck bunny	-	34.331	-	38.254	-	40.983
controlled burn	-	36.328	-	40.200	-	42.634
elephants dream	-	31.417	-	35.309	-	38.243
moving zone plate	-	31.290	-	35.281	-	38.060
speed bag	-	32.072	-	35.669	-	38.065
tractor	-	28.020	-	31.465	-	34.151
Moyenne	-	33.102	-	36.906	-	39.490

Tableau-A I-55 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 2000 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	37.275	7.973	63.515	14.075	80.900	20.491
big buck bunny	32.820	6.708	56.115	12.026	70.588	17.669
controlled burn	31.148	7.132	53.239	12.716	69.042	18.621
elephants dream	32.773	5.959	56.389	10.794	72.884	15.948
moving zone plate	30.791	5.927	53.749	10.737	74.913	15.878
speed bag	28.907	6.114	51.958	11.015	69.056	16.214
tractor	32.274	5.224	57.144	9.510	78.114	14.079
Moyenne	32.284	6.434	56.016	11.553	73.642	16.986

Tableau-A I-56 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 2000 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	94.944	25.103	103.927	31.306	-	34.840
big buck bunny	80.066	21.934	86.632	27.402	-	30.564
controlled burn	79.340	22.977	86.743	28.666	-	31.982
elephants dream	79.703	19.856	90.021	25.070	-	27.991
moving zone plate	84.031	19.824	102.399	24.814	-	27.832
speed bag	81.922	20.105	90.114	25.104	-	28.141
tractor	88.952	17.540	104.362	22.134	-	24.852
Moyenne	84.137	21.048	94.885	26.357	-	29.457

Tableau-A I-57 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 2000 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	**7 coeurs**		**8 coeurs**		**9 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	38.344	-	42.266	-	44.293
big buck bunny	-	33.771	-	37.678	-	40.328
controlled burn	-	35.317	-	39.131	-	41.523
elephants dream	-	31.182	-	35.077	-	37.871
moving zone plate	-	30.666	-	34.696	-	37.412
speed bag	-	31.128	-	34.643	-	37.007
tractor	-	27.645	-	31.135	-	33.745
Moyenne	-	32.579	-	36.375	-	38.883

Tableau-A I-58 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 4000 kb/s où le décodeur utilise 1, 2, et 3 coeurs DSP.

Sommaire	**1 coeur**		**2 coeurs**		**3 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	29.521	8.014	52.154	14.133	69.964	20.520
big buck bunny	23.743	6.386	43.521	11.481	59.109	16.897
controlled burn	21.614	6.471	39.737	11.620	54.704	17.054
elephants dream	22.332	5.756	41.104	10.431	56.413	15.410
moving zone plate	21.657	5.694	40.327	10.313	56.601	15.260
speed bag	19.485	5.563	36.455	10.069	51.443	14.852
tractor	21.904	5.023	40.495	9.148	55.697	13.550
Moyenne	22.894	6.130	41.970	11.028	57.704	16.220

Tableau-A I-59 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 4000 kb/s où le décodeur utilise 4, 5, et 6 coeurs DSP.

Sommaire	**4 coeurs**		**5 coeurs**		**6 coeurs**	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	82.443	25.213	92.858	31.258	-	34.860
big buck bunny	70.733	20.996	79.519	26.228	-	29.305
controlled burn	67.645	21.125	76.816	26.382	-	29.482
elephants dream	70.096	19.143	80.297	24.165	-	27.008
moving zone plate	71.133	18.966	81.840	23.749	-	26.528
speed bag	64.595	18.470	75.602	23.067	-	25.934
tractor	69.642	16.840	79.351	21.299	-	23.910
Moyenne	70.898	20.108	80.898	25.164	-	28.147

Tableau-A I-60 Vitesses d'exécution moyennes (fps) pour des séquences vidéo utilisant la résolution 720p et un débit d'encodage de 4000 kb/s où le décodeur utilise 7, 8, et 9 coeurs DSP.

Sommaire	7 coeurs		8 coeurs		9 coeurs	
	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)	f_1 (fps)	f_2 (fps)
bad apple	-	38.357	-	42.196	-	44.199
big buck bunny	-	32.392	-	36.145	-	38.790
controlled burn	-	32.562	-	36.328	-	38.695
elephants dream	-	29.981	-	33.794	-	36.501
moving zone plate	-	29.259	-	33.319	-	35.891
speed bag	-	28.670	-	31.952	-	34.307
tractor	-	26.670	-	29.927	-	32.376
Moyenne	-	31.127	-	34.809	-	37.251

4 Résultats d'accélérations du décodeur H.264/MPEG-4 AVC

Cette section a pour objectif de présenter les résultats obtenus pour l'accélération du décodeur H.264. La sous-section 4.1 présente les résultats obtenus pour la résolution 360p, alors que la sous-section 4.2 présente les résultats obtenus pour la résolution NTSC. Pour terminer, la sous-section 4.3 présente les résultats obtenus pour la résolution 720p et qui n'ont pas étés présentés au chapitre 5.

4.1 Résolution 360p

Les vitesses d'exécution moyennes en fps et l'accélération du décodeur H.264 pour des séquences vidéo encodées avec un débit de 512 kb/s, 768 kb/s, 1000 kb/s, et 2000 kb/s, utilisant la résolution 360p sont présentées dans cette sous-section. À l'intérieur des tableaux de cette sous-section, le paramètre f_1 représente le nombre de trames par secondes (fps) que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique, le paramètre f_2 représente le nombre de trames par secondes (fps) que peut traiter le processus de la reconstruction et du filtrage antibloc, le paramètre S représente le nombre de trames par secondes (fps) que peut traiter le système global du décodage H.264, c.-à-d. $1/(1/f_1 + 1/f_2)$ pour l'approche séquentielle et $min(f_1, f_2)$ pour les autres cas parallèles, et le paramètre A représente l'accélération qui est fournie par l'équation suivante :

Tableau-A I-61 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 512 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	144.124	30.318	24.990	1.000	144.124	30.318	30.318	1.213
big buck bunny	129.568	26.923	22.245	1.000	129.568	26.923	26.923	1.210
controlled burn	124.671	29.231	23.627	1.000	124.671	29.231	29.231	1.237
elephants dream	124.109	23.418	19.664	1.000	124.109	23.418	23.418	1.191
moving zone plate	120.028	23.425	19.564	1.000	120.028	23.425	23.425	1.197
speed bag	111.983	25.189	20.524	1.000	111.983	25.189	25.189	1.227
tractor	122.841	21.885	18.543	1.000	122.841	21.885	21.885	1.180
Moyenne	125.332	25.770	21.308	1.000	125.332	25.770	25.770	1.209

Tableau-A I-62 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 512 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Séquence	**Parallèle 1/2 coeurs DSP**				**Parallèle 1/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	142.645	49.190	49.190	1.968	144.124	69.751	69.751	2.791
big buck bunny	128.603	44.292	44.292	1.991	129.568	62.738	62.738	2.820
controlled burn	123.704	47.803	47.803	2.023	124.671	67.761	67.761	2.868
elephants dream	123.372	39.187	39.187	1.993	124.109	56.088	56.088	2.852
moving zone plate	119.495	39.120	39.120	2.000	120.028	56.313	56.313	2.878
speed bag	111.425	41.749	41.749	2.034	111.983	59.334	59.334	2.891
tractor	122.160	37.170	37.170	2.005	122.841	53.344	53.344	2.877
Moyenne	124.486	42.644	42.644	2.001	125.332	60.761	60.761	2.852

$$A = \frac{TempsSystèmeRéférence}{TempsSystèmeNouveau} = \frac{FpsSystèmeNouveau}{FpsSystèmeRéférence}, \tag{I.1}$$

dans lequel *Nouveau* est un système proposé à l'aide d'une configuration parallèle de coeurs DSP et *Référence* est le système de départ, soit le système avec une configuration séquentielle n'utilisant qu'un seul coeur DSP.

Tableau-A I-63 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 512 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 2/4.

Séquence	**Parallèle 1/4 coeurs DSP**				**Parallèle 2/4 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	144.124	88.133	88.133	3.527	189.871	88.133	88.133	3.527
big buck bunny	129.568	79.374	79.374	3.568	158.406	79.374	79.374	3.568
controlled burn	124.671	85.566	85.566	3.622	155.321	85.566	85.566	3.622
elephants dream	124.109	71.658	71.658	3.644	145.856	71.658	71.658	3.644
moving zone plate	120.028	71.457	71.457	3.652	141.151	71.457	71.457	3.652
speed bag	111.983	75.486	75.486	3.678	147.442	75.486	75.486	3.678
tractor	122.841	68.241	68.241	3.680	154.992	68.241	68.241	3.680
Moyenne	125.332	77.131	77.131	3.620	156.148	77.131	77.131	3.620

Tableau-A I-64 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 768 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	122.059	30.049	24.058	1.000	122.059	30.049	30.049	1.249
big buck bunny	107.297	26.112	20.960	1.000	107.297	26.112	26.112	1.246
controlled burn	101.841	27.718	21.744	1.000	101.841	27.718	27.718	1.275
elephants dream	101.167	22.905	18.644	1.000	101.167	22.905	22.905	1.229
moving zone plate	97.233	22.523	18.256	1.000	97.233	22.523	22.523	1.234
speed bag	89.485	24.054	18.924	1.000	89.485	24.054	24.054	1.271
tractor	99.563	21.405	17.588	1.000	99.563	21.405	21.405	1.217
Moyenne	102.664	24.967	20.025	1.000	102.664	24.967	24.967	1.247

Tableau-A I-65 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 768 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Séquence	**Parallèle 1/2 coeurs DSP**				**Parallèle 1/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	121.040	48.763	48.763	2.027	122.059	68.974	68.974	2.867
big buck bunny	106.707	43.068	43.068	2.055	107.297	60.928	60.928	2.907
controlled burn	101.238	45.679	45.679	2.101	101.841	64.710	64.710	2.976
elephants dream	100.741	38.341	38.341	2.056	101.167	54.757	54.757	2.937
moving zone plate	96.985	37.673	37.673	2.064	97.233	53.734	53.734	2.943
speed bag	89.165	40.050	40.050	2.116	89.485	56.861	56.861	3.005
tractor	99.162	36.365	36.365	2.068	99.563	51.870	51.870	2.949
Moyenne	102.148	41.420	41.420	2.068	102.664	58.833	58.833	2.938

Tableau-A I-66 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 768 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 2/4.

Séquence	**Parallèle 1/4 coeurs DSP**				**Parallèle 2/4 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	122.059	87.213	87.213	3.625	176.805	87.213	87.213	3.625
big buck bunny	107.297	77.077	77.077	3.677	152.621	77.077	77.077	3.677
controlled burn	101.841	81.818	81.818	3.763	143.322	81.818	81.818	3.763
elephants dream	101.167	69.875	69.875	3.748	148.483	69.875	69.875	3.748
moving zone plate	97.233	68.599	68.599	3.758	156.250	68.599	68.599	3.758
speed bag	89.485	72.387	72.387	3.825	132.565	72.387	72.387	3.825
tractor	99.563	66.443	66.443	3.778	160.422	66.443	66.443	3.778
Moyenne	102.664	74.773	74.773	3.734	152.924	74.773	74.773	3.734

Tableau-A I-67 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	108.884	29.900	23.406	1.000	108.884	29.900	29.900	1.277
big buck bunny	93.618	25.587	20.057	1.000	93.618	25.587	25.587	1.276
controlled burn	88.173	26.674	20.440	1.000	88.173	26.674	26.674	1.305
elephants dream	87.599	22.485	17.862	1.000	87.599	22.485	22.485	1.259
moving zone plate	83.727	21.940	17.356	1.000	83.727	21.940	21.940	1.264
speed bag	76.544	23.208	17.779	1.000	76.544	23.208	23.208	1.305
tractor	86.571	20.934	16.831	1.000	86.571	20.934	20.934	1.244
Moyenne	89.302	24.390	19.104	1.000	89.302	24.390	24.390	1.277

Tableau-A I-68 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Séquence	**Parallèle 1/2 coeurs DSP**				**Parallèle 1/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	108.030	48.553	48.553	2.074	108.884	68.354	68.354	2.920
big buck bunny	93.184	42.260	42.260	2.107	93.618	59.847	59.847	2.984
controlled burn	87.743	44.134	44.134	2.159	88.173	62.521	62.521	3.059
elephants dream	87.353	37.693	37.693	2.110	87.599	53.678	53.678	3.005
moving zone plate	83.540	36.768	36.768	2.118	83.727	52.308	52.308	3.014
speed bag	76.319	38.771	38.771	2.181	76.544	55.097	55.097	3.099
tractor	86.322	35.595	35.595	2.115	86.571	50.904	50.904	3.024
Moyenne	88.927	40.539	40.539	2.122	89.302	57.530	57.530	3.011

Tableau-A I-69 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 2/4.

Séquence	**Parallèle 1/4 coeurs DSP**				**Parallèle 2/4 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	108.884	86.589	86.589	3.699	167.897	86.589	86.589	3.699
big buck bunny	93.618	75.768	75.768	3.778	138.222	75.768	75.768	3.778
controlled burn	88.173	79.189	79.189	3.874	134.468	79.189	79.189	3.874
elephants dream	87.599	68.538	68.538	3.837	133.208	68.538	68.538	3.837
moving zone plate	83.727	67.031	67.031	3.862	134.253	67.031	67.031	3.862
speed bag	76.544	70.178	70.178	3.947	122.101	70.178	70.178	3.947
tractor	86.571	65.233	65.233	3.876	130.606	65.233	65.233	3.876
Moyenne	89.302	73.218	73.218	3.833	137.251	73.218	73.218	3.833

Tableau-A I-70 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	90.223	29.904	22.413	1.000	90.223	29.904	29.904	1.334
big buck bunny	63.937	23.847	17.341	1.000	63.937	23.847	23.847	1.375
controlled burn	57.702	23.569	16.708	1.000	57.702	23.569	23.569	1.411
elephants dream	58.475	21.018	15.439	1.000	58.475	21.018	21.018	1.361
moving zone plate	55.346	20.234	14.797	1.000	55.346	20.234	20.234	1.367
speed bag	50.138	20.918	14.740	1.000	50.138	20.918	20.918	1.419
tractor	57.542	19.484	14.535	1.000	57.542	19.484	19.484	1.340
Moyenne	61.909	22.711	16.568	1.000	61.909	22.711	22.711	1.371

Tableau-A I-71 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Séquence	**Parallèle 1/2 coeurs DSP**				**Parallèle 1/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	89.648	48.581	48.581	2.168	90.223	68.217	68.217	3.044
big buck bunny	63.746	39.631	39.631	2.285	63.937	56.233	56.233	3.243
controlled burn	57.559	39.604	39.604	2.370	57.702	55.847	55.847	3.343
elephants dream	58.382	35.406	35.406	2.293	58.475	50.287	50.287	3.257
moving zone plate	55.264	34.149	34.149	2.308	55.346	48.410	48.410	3.272
speed bag	50.049	35.249	35.249	2.391	50.138	50.171	50.138	3.401
tractor	57.453	33.272	33.272	2.289	57.542	47.559	47.559	3.272
Moyenne	61.729	37.985	37.985	2.293	61.909	53.818	53.813	3.248

Tableau-A I-72 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 2/4.

Séquence	**Parallèle 1/4 coeurs DSP**				**Parallèle 2/4 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	90.223	86.306	86.306	3.851	90.223	86.306	86.306	3.851
big buck bunny	63.937	71.378	63.937	3.687	107.035	71.378	71.378	4.116
controlled burn	57.702	71.168	57.702	3.454	98.940	71.168	71.168	4.260
elephants dream	58.475	64.333	58.475	3.787	100.733	64.333	64.333	4.167
moving zone plate	55.346	62.749	55.346	3.740	96.282	62.749	62.749	4.241
speed bag	50.138	64.138	50.138	3.401	87.878	64.138	64.138	4.351
tractor	57.542	61.135	57.542	3.959	99.125	61.135	61.135	4.206
Moyenne	61.909	68.744	61.349	3.703	97.174	68.744	68.744	4.149

Tableau-A I-73 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Sommaire	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
512 kb/s	125.332	25.770	21.308	1.000	125.332	25.770	25.770	1.209
768 kb/s	102.664	24.967	20.025	1.000	102.664	24.967	24.967	1.247
1000 kb/s	89.302	24.390	19.104	1.000	89.302	24.390	24.390	1.277
2000 kb/s	61.909	22.711	16.568	1.000	61.909	22.711	22.711	1.371
Moyenne	94.802	24.459	19.251	1.000	94.802	24.459	24.459	1.276

Tableau-A I-74 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 2/3.

Sommaire	**Parallèle 1/2 coeurs DSP**				**Parallèle 2/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
512 kb/s	124.486	42.644	42.644	2.001	125.332	60.761	60.761	2.852
768 kb/s	102.148	41.420	41.420	2.068	102.664	58.833	58.833	2.938
1000 kb/s	88.927	40.539	40.539	2.122	89.302	57.530	57.530	3.011
2000 kb/s	61.729	37.985	37.985	2.293	61.909	53.818	53.813	3.248
Moyenne	94.323	40.647	40.647	2.121	94.802	57.736	57.734	3.012

Tableau-A I-75 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 360p où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 2/4.

Sommaire	Parallèle 1/4 coeurs DSP				Parallèle 2/4 coeurs DSP			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
512 kb/s	125.332	77.131	77.131	3.620	156.148	77.131	77.131	3.620
768 kb/s	102.664	74.773	74.773	3.734	152.924	74.773	74.773	3.734
1000 kb/s	89.302	73.218	73.218	3.833	137.251	73.218	73.218	3.833
2000 kb/s	61.909	68.744	61.349	3.703	97.174	68.744	68.744	4.149
Moyenne	94.802	73.466	71.618	3.722	135.874	73.466	73.466	3.834

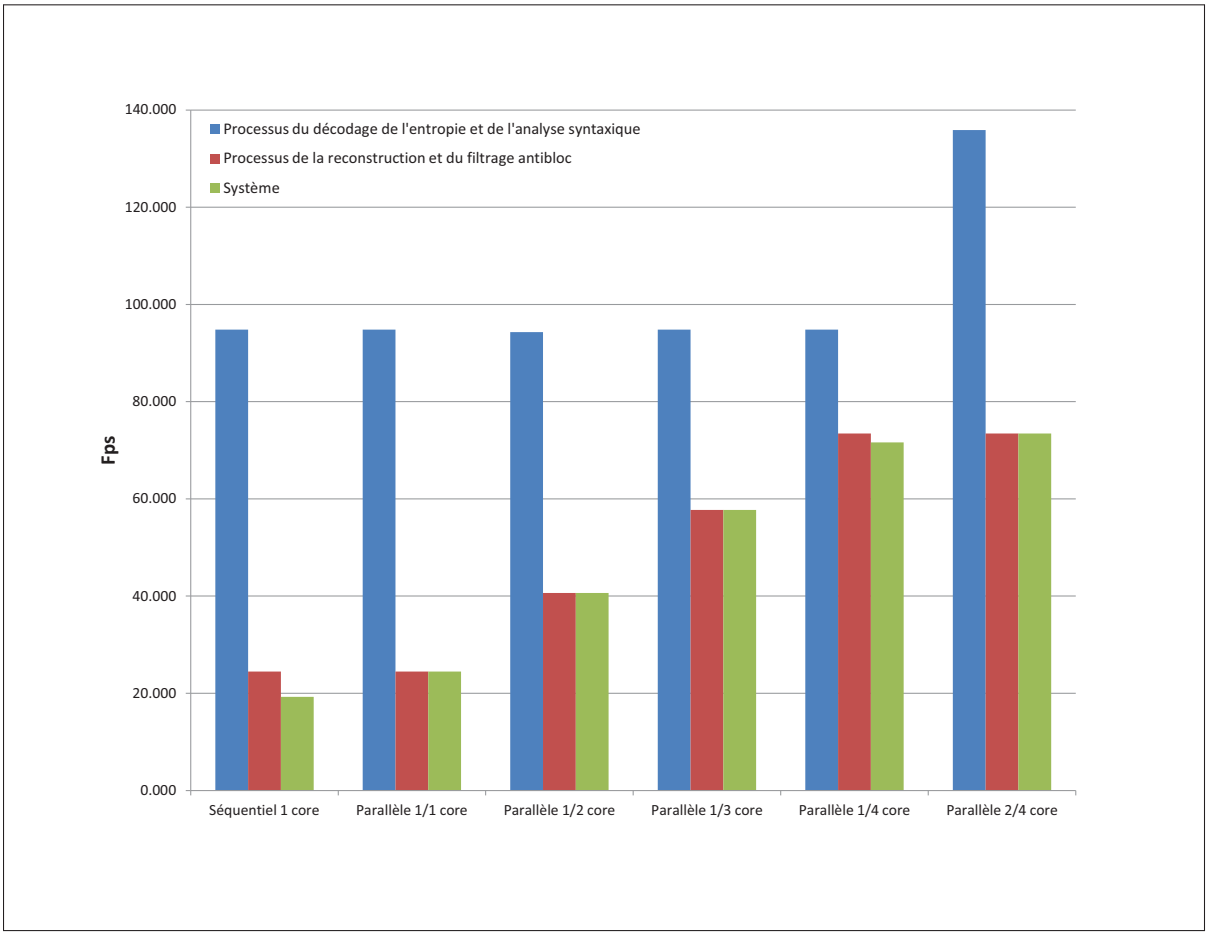


Figure-A I-7 Histogramme des vitesses d'exécution moyennes pour différentes configurations de coeurs DSP pour la résolution 360p.

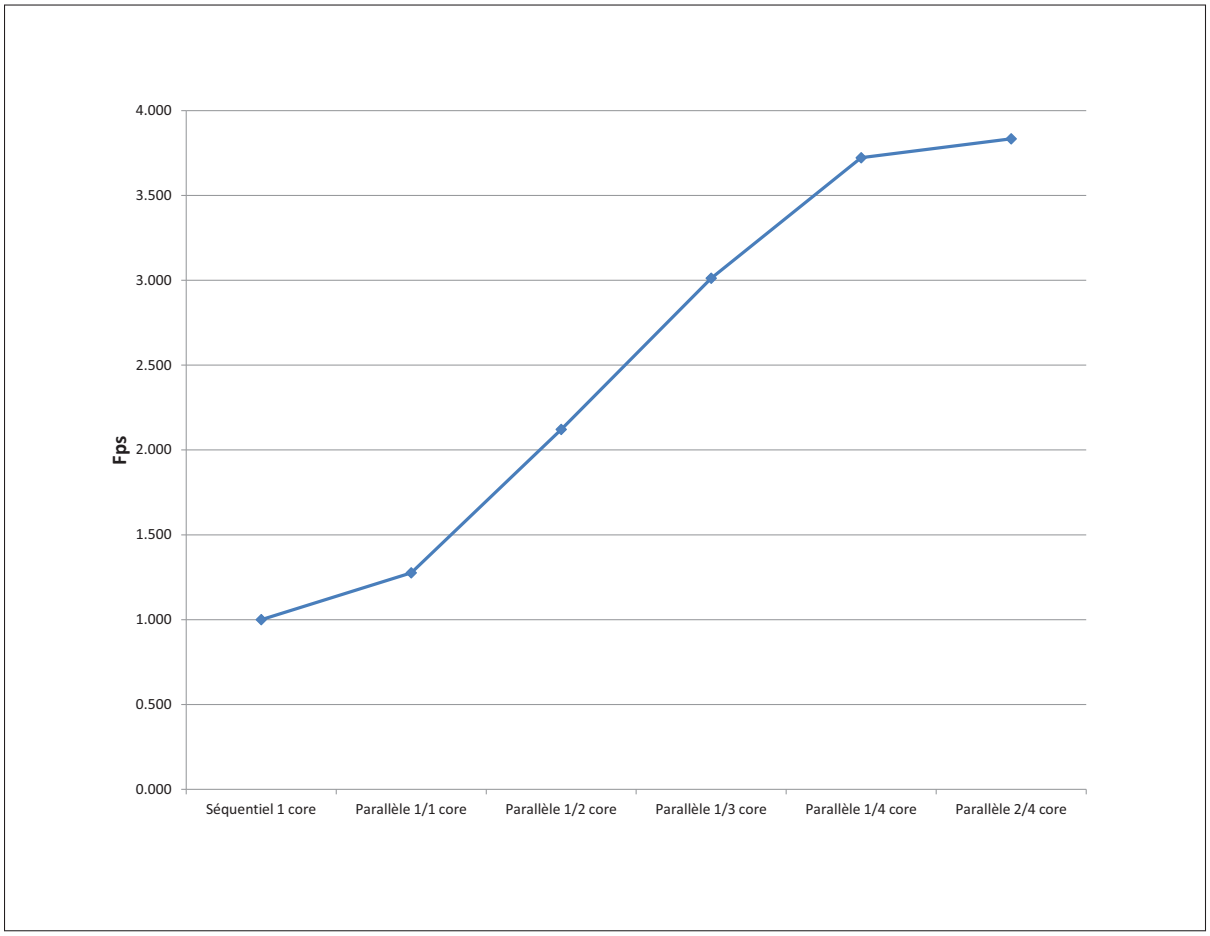


Figure-A I-8 Courbe d'accélération en fonction de différentes configurations de coeurs DSP pour la résolution 360p.

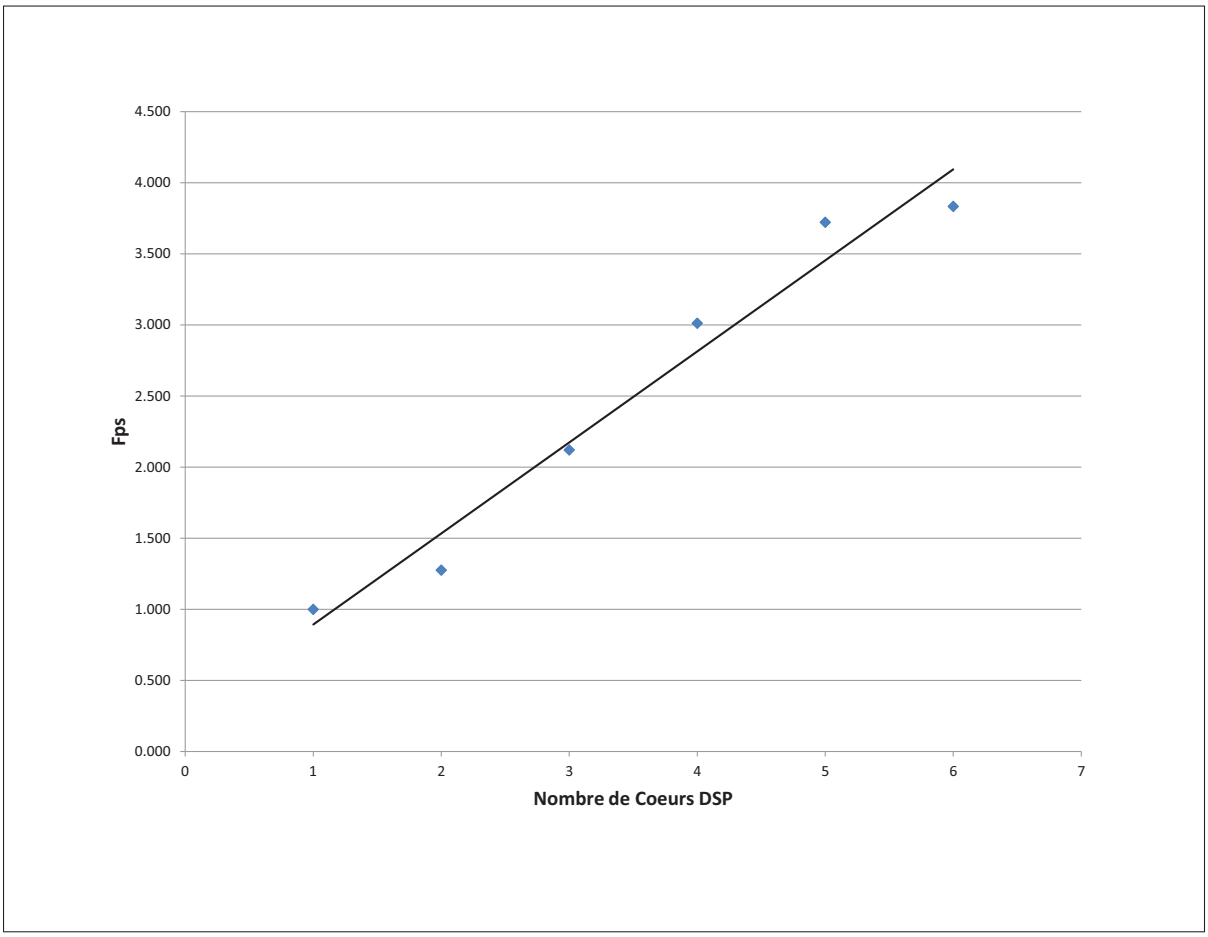


Figure-A I-9 Courbe d'accélération en fonction du nombre de coeurs DSP pour la résolution 360p.

4.2 Résolution NTSC

Les vitesses d'exécution moyennes en fps et l'accélération du décodeur H.264 pour des séquences vidéo encodées avec un débit de 512 kb/s, 768 kb/s, 1000 kb/s, et 2000 kb/s, utilisant la résolution NTSC sont présentées dans cette sous-section. À l'intérieur des tableaux de cette sous-section, le paramètre f_1 représente le nombre de trames par secondes (fps) que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique, le paramètre f_2 représente le nombre de trames par secondes (fps) que peut traiter le processus de la reconstruction et du filtrage antibloc, le paramètre S représente le nombre de trames par secondes (fps) que peut traiter le système global du décodage H.264, c.-à-d. $1/(1/f_1 + 1/f_2)$ pour l'approche séquen-

Tableau-A I-76 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 512 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	Séquentiel 1 coeur DSP				Parallèle 1/1 coeurs DSP			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	111.849	20.698	17.437	1.000	111.849	20.698	20.698	1.187
big buck bunny	102.661	18.813	15.875	1.000	102.661	18.813	18.813	1.185
controlled burn	99.801	20.651	17.083	1.000	99.801	20.651	20.651	1.209
elephants dream	101.222	16.117	13.885	1.000	101.222	16.117	16.117	1.161
moving zone plate	98.971	16.333	14.001	1.000	98.971	16.333	16.333	1.167
speed bag	92.207	17.641	14.787	1.000	92.207	17.641	17.641	1.193
tractor	101.485	14.901	12.977	1.000	101.485	14.901	14.901	1.148
Moyenne	101.171	17.879	15.149	1.000	101.171	17.879	17.879	1.180

Tableau-A I-77 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 512 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Séquence	Parallèle 1/2 coeurs DSP				Parallèle 1/3 coeurs DSP			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	110.524	35.016	35.016	2.008	111.849	49.924	49.924	2.863
big buck bunny	101.731	32.175	32.175	2.027	102.661	46.145	46.145	2.907
controlled burn	98.794	34.915	34.915	2.044	99.801	49.813	49.813	2.916
elephants dream	100.392	28.092	28.092	2.023	101.222	40.706	40.706	2.932
moving zone plate	98.280	28.432	28.432	2.031	98.971	41.198	41.198	2.943
speed bag	91.626	30.332	30.332	2.051	92.207	43.467	43.467	2.940
tractor	100.654	26.094	26.094	2.011	101.485	37.830	37.830	2.915
Moyenne	100.286	30.722	30.722	2.028	101.171	44.155	44.155	2.915

tielle et $min(f_1, f_2)$ pour les autres cas parallèles, et le paramètre A représente l'accélération qui est fournie par l'équation suivante :

$$A = \frac{TempsSystèmeRéférence}{TempsSystèmeNouveau} = \frac{FpsSystèmeNouveau}{FpsSystèmeRéférence}, \tag{I.2}$$

dans lequel *Nouveau* est un système proposé à l'aide d'une configuration parallèle de coeurs DSP et *Référence* est le système de départ, soit le système avec une configuration séquentielle n'utilisant qu'un seul coeur DSP.

Tableau-A I-78 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 512 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 1/5.

Séquence	**Parallèle 1/4 coeurs DSP**				**Parallèle 1/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	111.849	61.861	61.861	3.548	111.849	72.956	72.956	4.184
big buck bunny	102.661	57.335	57.335	3.612	102.661	67.950	67.950	4.280
controlled burn	99.801	61.588	61.588	3.605	99.801	72.657	72.657	4.253
elephants dream	101.222	50.829	50.829	3.661	101.222	60.858	60.858	4.383
moving zone plate	98.971	51.520	51.520	3.680	98.971	61.518	61.518	4.394
speed bag	92.207	54.023	54.023	3.653	92.207	64.112	64.112	4.336
tractor	101.485	47.449	47.449	3.656	101.485	56.928	56.928	4.387
Moyenne	101.171	54.944	54.944	3.627	101.171	65.283	65.283	4.309

Tableau-A I-79 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 768 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	96.903	20.647	16.993	1.000	96.903	20.647	20.647	1.215
big buck bunny	87.307	18.220	15.053	1.000	87.307	18.220	18.220	1.210
controlled burn	83.529	19.654	15.887	1.000	83.529	19.654	19.654	1.237
elephants dream	84.511	15.895	13.362	1.000	84.511	15.895	15.895	1.190
moving zone plate	80.824	15.961	13.312	1.000	80.824	15.961	15.961	1.199
speed bag	75.636	16.950	13.829	1.000	75.636	16.950	16.950	1.226
tractor	83.572	14.687	12.477	1.000	83.572	14.687	14.687	1.177
Moyenne	84.612	17.431	14.416	1.000	84.612	17.431	17.431	1.209

Tableau-A I-80 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 768 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Séquence	**Parallèle 1/2 coeurs DSP**				**Parallèle 1/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	95.935	34.930	34.930	2.056	96.903	49.700	49.700	2.925
big buck bunny	86.691	31.265	31.265	2.077	87.307	44.844	44.844	2.979
controlled burn	82.916	33.425	33.425	2.104	83.529	47.685	47.685	3.002
elephants dream	84.020	27.715	27.715	2.074	84.511	40.102	40.102	3.001
moving zone plate	80.449	27.792	27.792	2.088	80.824	40.228	40.228	3.022
speed bag	75.267	29.244	29.244	2.115	75.636	41.944	41.944	3.033
tractor	83.107	25.765	25.765	2.065	83.572	37.289	37.289	2.989
Moyenne	84.055	30.019	30.019	2.082	84.612	43.113	43.113	2.991

Tableau-A I-81 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 768 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 1/5.

Séquence	**Parallèle 1/4 coeurs DSP**				**Parallèle 1/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	96.903	61.396	61.396	3.613	96.903	72.497	72.497	4.266
big buck bunny	87.307	55.513	55.513	3.688	87.307	65.965	65.965	4.382
controlled burn	83.529	59.089	59.089	3.719	83.529	69.717	69.717	4.388
elephants dream	84.511	49.852	49.852	3.731	84.511	59.951	59.951	4.487
moving zone plate	80.824	49.801	49.801	3.741	80.824	60.052	60.052	4.511
speed bag	75.636	51.981	51.981	3.759	75.636	61.866	61.866	4.474
tractor	83.572	46.267	46.267	3.708	83.572	56.071	56.071	4.494
Moyenne	84.612	53.414	53.414	3.705	84.612	63.731	63.731	4.421

Tableau-A I-82 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	87.083	20.547	16.599	1.000	87.083	20.547	20.547	1.238
big buck bunny	77.131	17.829	14.462	1.000	77.131	17.829	17.829	1.233
controlled burn	73.351	19.003	15.071	1.000	73.351	19.003	19.003	1.261
elephants dream	74.326	15.705	12.950	1.000	74.326	15.705	15.705	1.213
moving zone plate	69.695	15.635	12.755	1.000	69.695	15.635	15.635	1.226
speed bag	65.515	16.454	13.135	1.000	65.515	16.454	16.454	1.253
tractor	73.108	14.463	12.061	1.000	73.108	14.463	14.463	1.199
Moyenne	74.316	17.091	13.862	1.000	74.316	17.091	17.091	1.233

Tableau-A I-83 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Séquence	**Parallèle 1/2 coeurs DSP**				**Parallèle 1/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	86.321	34.735	34.735	2.093	87.083	49.382	49.382	2.975
big buck bunny	76.666	30.662	30.662	2.120	77.131	43.990	43.990	3.042
controlled burn	72.903	32.448	32.448	2.153	73.351	46.295	46.295	3.072
elephants dream	73.979	27.423	27.423	2.118	74.326	39.635	39.635	3.061
moving zone plate	69.465	27.253	27.253	2.137	69.695	39.408	39.408	3.090
speed bag	65.253	28.437	28.437	2.165	65.515	40.801	40.801	3.106
tractor	72.785	25.397	25.397	2.106	73.108	36.764	36.764	3.048
Moyenne	73.910	29.479	29.479	2.127	74.316	42.325	42.325	3.053

Tableau-A I-84 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 1/5.

Séquence	**Parallèle 1/4 coeurs DSP**				**Parallèle 1/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	87.083	60.955	60.955	3.672	87.083	72.097	72.097	4.343
big buck bunny	77.131	54.448	54.448	3.765	77.131	64.737	64.737	4.476
controlled burn	73.351	57.303	57.303	3.802	73.351	67.840	67.840	4.501
elephants dream	74.326	49.240	49.240	3.802	74.326	59.297	59.297	4.579
moving zone plate	69.695	48.746	48.746	3.822	69.695	58.666	58.666	4.599
speed bag	65.515	50.601	50.601	3.852	65.515	60.225	60.225	4.585
tractor	73.108	45.611	45.611	3.782	73.108	55.308	55.308	4.586
Moyenne	74.316	52.415	52.415	3.781	74.316	62.596	62.596	4.516

Tableau-A I-85 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	67.226	20.467	15.667	1.000	67.226	20.467	20.467	1.306
big buck bunny	54.273	16.830	12.831	1.000	54.273	16.830	16.830	1.312
controlled burn	49.593	17.062	12.680	1.000	49.593	17.062	17.062	1.346
elephants dream	50.730	15.012	11.571	1.000	50.730	15.012	15.012	1.297
moving zone plate	46.440	14.609	11.101	1.000	46.440	14.609	14.609	1.316
speed bag	43.302	14.931	11.091	1.000	43.302	14.931	14.931	1.346
tractor	49.489	13.695	10.716	1.000	49.489	13.695	13.695	1.278
Moyenne	51.579	16.087	12.237	1.000	51.579	16.087	16.087	1.315

Tableau-A I-86 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Séquence	**Parallèle 1/2 coeurs DSP**				**Parallèle 1/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	66.764	34.572	34.572	2.207	67.226	49.167	49.167	3.138
big buck bunny	54.078	29.110	29.110	2.269	54.273	41.799	41.799	3.258
controlled burn	49.424	29.424	29.424	2.321	49.593	42.115	42.115	3.321
elephants dream	50.611	26.289	26.289	2.272	50.730	37.933	37.933	3.278
moving zone plate	46.369	25.527	25.527	2.300	46.440	36.894	36.894	3.323
speed bag	43.220	26.041	26.041	2.348	43.302	37.429	37.429	3.375
tractor	49.372	24.140	24.140	2.253	49.489	34.937	34.937	3.260
Moyenne	51.405	27.872	27.872	2.278	51.579	40.039	40.039	3.272

Tableau-A I-87 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 1/5.

Séquence	**Parallèle 1/4 coeurs DSP**				**Parallèle 1/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	67.226	60.661	60.661	3.872	67.226	71.452	67.226	4.291
big buck bunny	54.273	51.748	51.748	4.033	54.273	61.617	54.273	4.230
controlled burn	49.593	52.151	49.593	3.911	49.593	62.015	49.593	3.911
elephants dream	50.730	47.051	47.051	4.066	50.730	56.834	50.730	4.384
moving zone plate	46.440	45.875	45.875	4.133	46.440	54.403	46.440	4.183
speed bag	43.302	46.503	43.302	3.904	43.302	55.438	43.302	3.904
tractor	49.489	43.413	43.413	4.051	49.489	52.573	49.489	4.618
Moyenne	51.579	49.629	48.806	3.989	51.579	59.190	51.579	4.215

Tableau-A I-88 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Sommaire	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
512 kb/s	101.171	17.879	15.149	1.000	101.171	17.879	17.879	1.180
768 kb/s	84.612	17.431	14.416	1.000	84.612	17.431	17.431	1.209
1000 kb/s	74.316	17.091	13.862	1.000	74.316	17.091	17.091	1.233
2000 kb/s	51.579	16.087	12.237	1.000	51.579	16.087	16.087	1.315
Moyenne	77.919	17.122	13.916	1.000	77.919	17.122	17.122	1.234

Tableau-A I-89 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC où le décodeur utilise une configuration de coeurs DSP parallèle 1/2 et parallèle 1/3.

Sommaire	**Parallèle 1/2 coeurs DSP**				**Parallèle 1/3 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
512 kb/s	100.286	30.722	30.722	2.028	101.171	44.155	44.155	2.915
768 kb/s	84.055	30.019	30.019	2.082	84.612	43.113	43.113	2.991
1000 kb/s	73.910	29.479	29.479	2.127	74.316	42.325	42.325	3.053
2000 kb/s	51.405	27.872	27.872	2.278	51.579	40.039	40.039	3.272
Moyenne	77.414	29.523	29.523	2.129	77.919	42.408	42.408	3.058

Tableau-A I-90 Vitesses d'exécution moyennes (fps) et accélération pour la résolution NTSC où le décodeur utilise une configuration de coeurs DSP parallèle 1/4 et parallèle 1/5.

Sommaire	**Parallèle 1/4 coeurs DSP**				**Parallèle 1/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
512 kb/s	101.171	54.944	54.944	3.627	101.171	65.283	65.283	4.309
768 kb/s	84.612	53.414	53.414	3.705	84.612	63.731	63.731	4.421
1000 kb/s	74.316	52.415	52.415	3.781	74.316	62.596	62.596	4.516
2000 kb/s	51.579	49.629	48.806	3.989	51.579	59.190	51.579	4.215
Moyenne	77.919	52.600	52.395	3.775	77.919	62.700	60.797	4.365

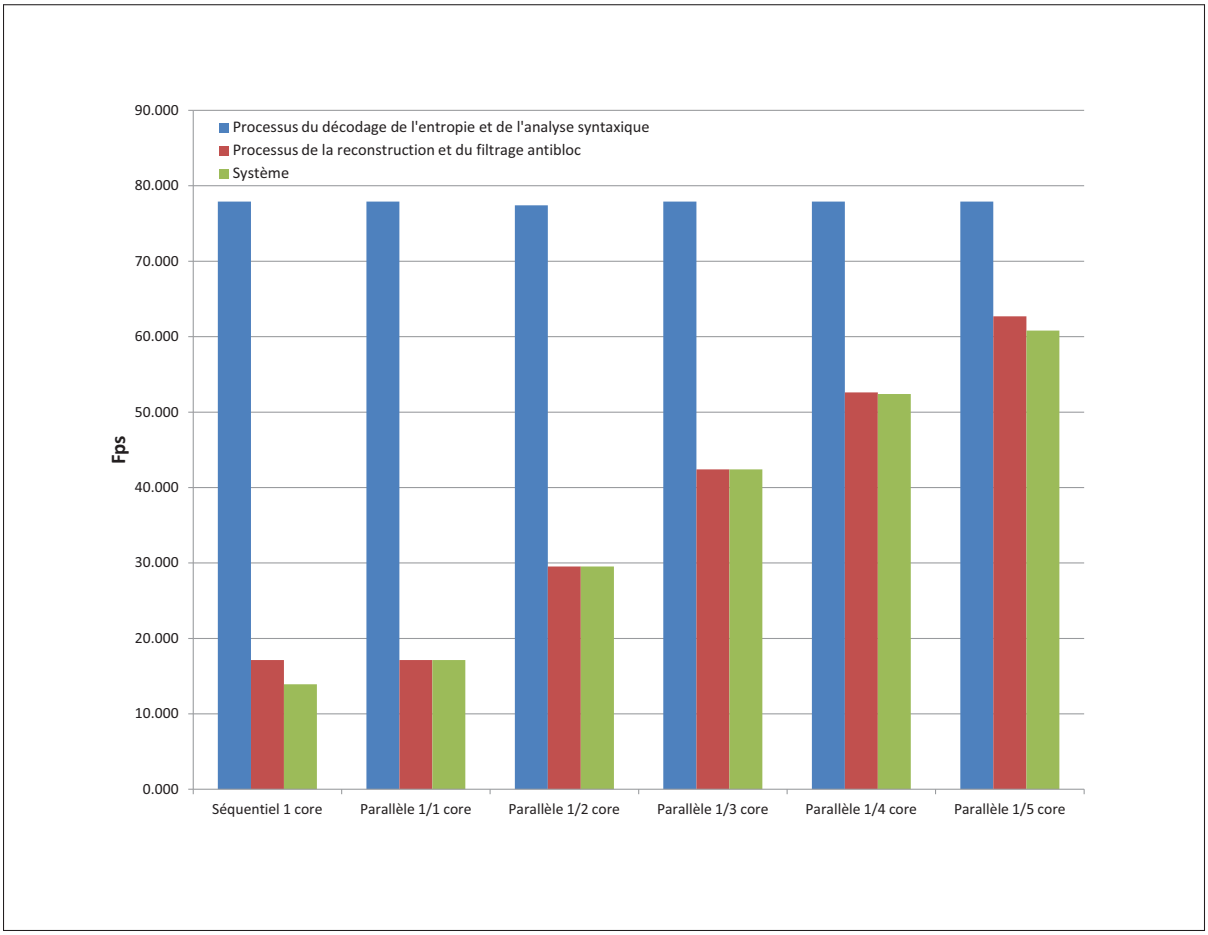


Figure-A I-10 Histogramme des vitesses d'exécution moyennes pour différentes configurations de coeurs DSP pour la résolution NTSC.

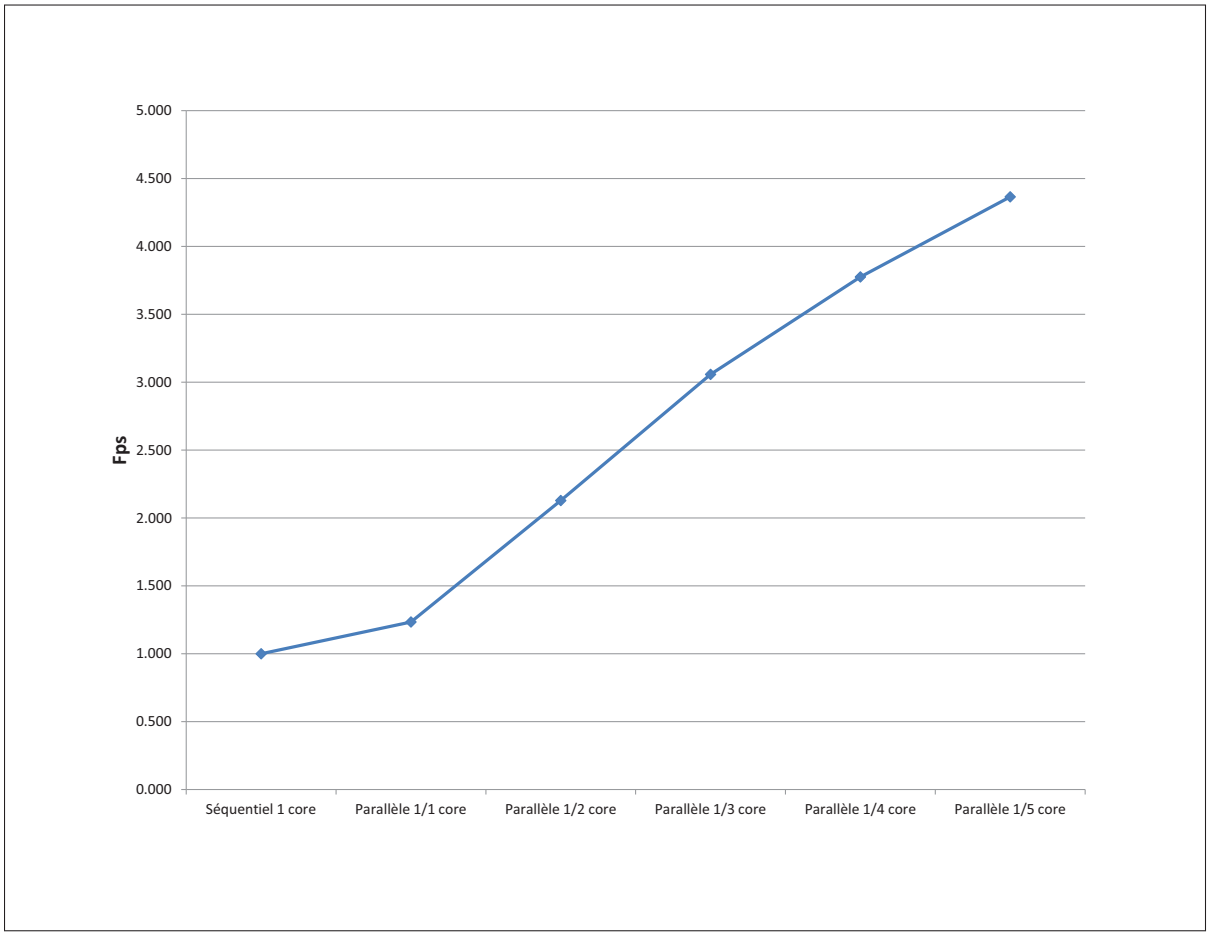


Figure-A I-11 Courbe d'accélération en fonction de différentes configuration de coeurs DSP pour la résolution NTSC.

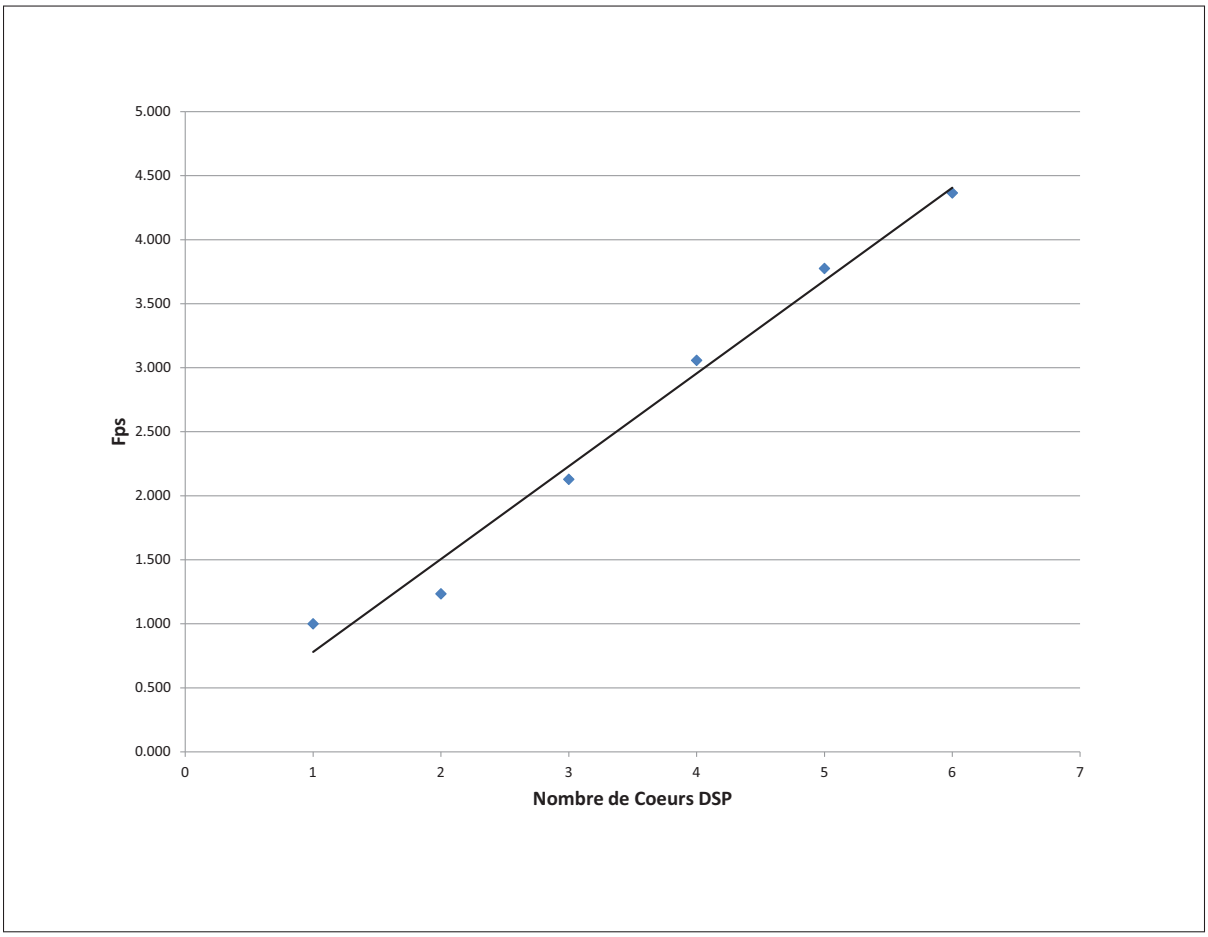


Figure-A I-12 Courbe d'accélération en fonction du nombre de coeurs DSP pour la résolution NTSC.

4.3 Résolution 720p

Les vitesses d'exécution moyennes en fps et l'accélération du décodeur H.264 pour des séquences vidéo encodées avec un débit de 1000 kb/s, 1500 kb/s, 2000 kb/s, et 4000 kb/s, utilisant la résolution 720p sont présentées dans cette sous-section. À l'intérieur des tableaux de cette sous-section, le paramètre f_1 représente le nombre de trames par secondes (fps) que peut traiter le processus du décodage de l'entropie et de l'analyse syntaxique, le paramètre f_2 représente le nombre de trames par secondes (fps) que peut traiter le processus de la reconstruction et du filtrage antibloc, le paramètre S représente le nombre de trames par secondes (fps) que peut traiter le système global du décodage H.264, c.-à-d. $1/(1/f_1 + 1/f_2)$ pour l'approche séquen-

Tableau-A I-91 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	Séquentiel 1 coeur DSP				Parallèle 1/1 coeurs DSP			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	46.528	7.946	6.783	1.000	46.528	7.946	7.946	1.171
big buck bunny	43.013	7.045	6.050	1.000	43.013	7.045	7.045	1.164
controlled burn	41.703	7.748	6.530	1.000	41.703	7.748	7.748	1.187
elephants dream	43.506	6.032	5.295	1.000	43.506	6.032	6.032	1.139
moving zone plate	42.621	6.054	5.298	1.000	42.621	6.054	6.054	1.143
speed bag	39.517	6.593	5.648	1.000	39.517	6.593	6.593	1.167
tractor	44.302	5.338	4.762	1.000	44.302	5.338	5.338	1.121
Moyenne	43.027	6.679	5.767	1.000	43.027	6.679	6.679	1.158

Tableau-A I-92 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 2/3 et parallèle 2/5.

Séquence	Parallèle 2/3 coeurs DSP				Parallèle 2/5 coeurs DSP			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	73.537	20.488	20.488	3.020	73.537	31.436	31.436	4.635
big buck bunny	64.569	18.490	18.490	3.056	64.569	28.691	28.691	4.742
controlled burn	61.403	20.065	20.065	3.073	61.403	30.883	30.883	4.729
elephants dream	66.461	16.176	16.176	3.055	66.461	25.447	25.447	4.806
moving zone plate	67.052	16.207	16.207	3.059	67.052	25.445	25.445	4.803
speed bag	63.341	17.372	17.372	3.076	63.341	26.898	26.898	4.762
tractor	73.444	14.401	14.401	3.024	73.444	22.652	22.652	4.757
Moyenne	67.115	17.600	17.600	3.052	67.115	27.350	27.350	4.743

tielle et $min(f_1, f_2)$ pour les autres cas parallèles, et le paramètre A représente l'accélération qui est fournie par l'équation suivante :

$$A = \frac{TempsSystèmeRéférence}{TempsSystèmeNouveau} = \frac{FpsSystèmeNouveau}{FpsSystèmeRéférence}, \tag{I.3}$$

dans lequel *Nouveau* est un système proposé à l'aide d'une configuration parallèle de coeurs DSP et *Référence* est le système de départ, soit le système avec une configuration séquentielle n'utilisant qu'un seul coeur DSP.

Tableau-A I-93 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 1000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 2/7 et parallèle 2/9.

Séquence	**Parallèle 2/7 coeurs DSP**				**Parallèle 2/9 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	73.537	38.542	38.542	5.682	73.537	44.654	44.654	6.583
big buck bunny	64.569	35.347	35.347	5.842	64.569	41.959	41.959	6.935
controlled burn	61.403	37.771	37.771	5.784	61.403	44.120	44.120	6.757
elephants dream	66.461	31.792	31.792	6.004	66.461	38.560	38.560	7.282
moving zone plate	67.052	31.681	31.681	5.980	67.052	38.525	38.525	7.272
speed bag	63.341	33.305	33.305	5.897	63.341	39.357	39.357	6.968
tractor	73.444	28.407	28.407	5.965	73.444	34.581	34.581	7.262
Moyenne	67.115	33.835	33.835	5.867	67.115	40.251	40.251	6.980

Tableau-A I-94 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 1500 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	41.154	7.940	6.651	1.000	41.154	7.940	7.940	1.194
big buck bunny	37.077	6.829	5.764	1.000	37.077	6.829	6.829	1.185
controlled burn	35.494	7.377	6.104	1.000	35.494	7.377	7.377	1.209
elephants dream	37.168	5.994	5.159	1.000	37.168	5.994	5.994	1.162
moving zone plate	35.565	6.003	5.133	1.000	35.565	6.003	6.003	1.169
speed bag	33.298	6.321	5.310	1.000	33.298	6.321	6.321	1.190
tractor	37.199	5.281	4.622	1.000	37.199	5.281	5.281	1.143
Moyenne	36.708	6.535	5.535	1.000	36.708	6.535	6.535	1.181

Tableau-A I-95 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 1500 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 2/3 et parallèle 2/5.

Séquence	**Parallèle 2/3 coeurs DSP**				**Parallèle 2/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	68.325	20.415	20.415	3.069	68.325	31.219	31.219	4.694
big buck bunny	60.341	17.968	17.968	3.117	60.341	27.880	27.880	4.837
controlled burn	58.595	19.198	19.198	3.145	58.595	29.581	29.581	4.846
elephants dream	62.551	16.053	16.053	3.112	62.551	25.242	25.242	4.893
moving zone plate	62.241	16.079	16.079	3.132	62.241	25.171	25.171	4.904
speed bag	57.755	16.713	16.713	3.147	57.755	25.869	25.869	4.872
tractor	65.018	14.227	14.227	3.078	65.018	22.369	22.369	4.840
Moyenne	62.118	17.236	17.236	3.114	62.118	26.762	26.762	4.835

Tableau-A I-96 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 1500 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 2/7 et parallèle 2/9.

Séquence	**Parallèle 2/7 coeurs DSP**				**Parallèle 2/9 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	68.325	38.257	38.257	5.752	68.325	44.297	44.297	6.660
big buck bunny	60.341	34.331	34.331	5.956	60.341	40.983	40.983	7.110
controlled burn	58.595	36.328	36.328	5.952	58.595	42.634	42.634	6.985
elephants dream	62.551	31.417	31.417	6.090	62.551	38.243	38.243	7.413
moving zone plate	62.241	31.290	31.290	6.096	62.241	38.060	38.060	7.415
speed bag	57.755	32.072	32.072	6.040	57.755	38.065	38.065	7.169
tractor	65.018	28.020	28.020	6.062	65.018	34.151	34.151	7.389
Moyenne	62.118	33.102	33.102	5.981	62.118	39.490	39.490	7.135

Tableau-A I-97 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	37.275	7.973	6.564	1.000	37.275	7.973	7.973	1.215
big buck bunny	32.820	6.708	5.567	1.000	32.820	6.708	6.708	1.205
controlled burn	31.148	7.132	5.800	1.000	31.148	7.132	7.132	1.230
elephants dream	32.773	5.959	5.040	1.000	32.773	5.959	5.959	1.182
moving zone plate	30.791	5.927	4.968	1.000	30.791	5.927	5.927	1.193
speed bag	28.907	6.114	5.044	1.000	28.907	6.114	6.114	1.212
tractor	32.274	5.224	4.495	1.000	32.274	5.224	5.224	1.162
Moyenne	32.284	6.434	5.354	1.000	32.284	6.434	6.434	1.202

Tableau-A I-98 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 2/3 et parallèle 2/5.

Séquence	**Parallèle 2/3 coeurs DSP**				**Parallèle 2/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	63.515	20.491	20.491	3.122	63.515	31.306	31.306	4.769
big buck bunny	56.115	17.669	17.669	3.174	56.115	27.402	27.402	4.922
controlled burn	53.239	18.621	18.621	3.211	53.239	28.666	28.666	4.942
elephants dream	56.389	15.948	15.948	3.164	56.389	25.070	25.070	4.974
moving zone plate	53.749	15.878	15.878	3.196	53.749	24.814	24.814	4.995
speed bag	51.958	16.214	16.214	3.215	51.958	25.104	25.104	4.977
tractor	57.144	14.079	14.079	3.132	57.144	22.134	22.134	4.924
Moyenne	56.016	16.986	16.986	3.173	56.016	26.357	26.357	4.923

Tableau-A I-99 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 2000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 2/7 et parallèle 2/9.

Séquence	**Parallèle 2/7 coeurs DSP**				**Parallèle 2/9 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	63.515	38.344	38.344	5.842	63.515	44.293	44.293	6.748
big buck bunny	56.115	33.771	33.771	6.066	56.115	40.328	40.328	7.244
controlled burn	53.239	35.317	35.317	6.089	53.239	41.523	41.523	7.159
elephants dream	56.389	31.182	31.182	6.187	56.389	37.871	37.871	7.514
moving zone plate	53.749	30.666	30.666	6.173	53.749	37.412	37.412	7.531
speed bag	51.958	31.128	31.128	6.171	51.958	37.007	37.007	7.337
tractor	57.144	27.645	27.645	6.150	57.144	33.745	33.745	7.507
Moyenne	56.016	32.579	32.579	6.085	56.016	38.883	38.883	7.262

Tableau-A I-100 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 4000 kb/s où le décodeur utilise une configuration de coeurs DSP séquentielle et parallèle 1/1.

Séquence	**Séquentiel 1 coeur DSP**				**Parallèle 1/1 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	29.521	8.014	6.299	1.000	29.521	8.014	8.014	1.272
big buck bunny	23.743	6.386	5.030	1.000	23.743	6.386	6.386	1.270
controlled burn	21.614	6.471	4.978	1.000	21.614	6.471	6.471	1.300
elephants dream	22.332	5.756	4.574	1.000	22.332	5.756	5.756	1.258
moving zone plate	21.657	5.694	4.507	1.000	21.657	5.694	5.694	1.263
speed bag	19.485	5.563	4.326	1.000	19.485	5.563	5.563	1.286
tractor	21.904	5.023	4.084	1.000	21.904	5.023	5.023	1.230
Moyenne	22.894	6.130	4.828	1.000	22.894	6.130	6.130	1.270

Tableau-A I-101 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 4000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 2/3 et parallèle 2/5.

Séquence	**Parallèle 2/3 coeurs DSP**				**Parallèle 2/5 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	52.154	20.520	20.520	3.258	52.154	31.258	31.258	4.962
big buck bunny	43.521	16.897	16.897	3.359	43.521	26.228	26.228	5.214
controlled burn	39.737	17.054	17.054	3.426	39.737	26.382	26.382	5.300
elephants dream	41.104	15.410	15.410	3.369	41.104	24.165	24.165	5.283
moving zone plate	40.327	15.260	15.260	3.386	40.327	23.749	23.749	5.269
speed bag	36.455	14.852	14.852	3.433	36.455	23.067	23.067	5.332
tractor	40.495	13.550	13.550	3.318	40.495	21.299	21.299	5.215
Moyenne	41.970	16.220	16.220	3.359	41.970	25.164	25.164	5.212

Tableau-A I-102 Vitesses d'exécution moyennes (fps) et accélération pour la résolution 720p pour des séquences vidéo encodées avec un débit de 4000 kb/s où le décodeur utilise une configuration de coeurs DSP parallèle 2/7 et parallèle 2/9.

Séquence	**Parallèle 2/7 coeurs DSP**				**Parallèle 2/9 coeurs DSP**			
	f_1 (fps)	f_2 (fps)	S (fps)	A	f_1 (fps)	f_2 (fps)	S (fps)	A
bad apple	52.154	38.357	38.357	6.089	52.154	44.199	44.199	7.017
big buck bunny	43.521	32.392	32.392	6.440	43.521	38.790	38.790	7.712
controlled burn	39.737	32.562	32.562	6.541	39.737	38.695	38.695	7.773
elephants dream	41.104	29.981	29.981	6.555	41.104	36.501	36.501	7.980
moving zone plate	40.327	29.259	29.259	6.492	40.327	35.891	35.891	7.963
speed bag	36.455	28.670	28.670	6.627	36.455	34.307	34.307	7.930
tractor	40.495	26.670	26.670	6.530	40.495	32.376	32.376	7.928
Moyenne	41.970	31.127	31.127	6.447	41.970	37.251	37.251	7.715

BIBLIOGRAPHIE

November 2007. *ITU-T Recommendation H.264 : Advanced video coding for generic audio-visual services*.

Arnold et Associates. 2010. « New Directions in Media Gateway Design ». White Paper.

Awad. 2011. « Affordable Video Conferencing Primed For Takeoff ». White Paper.

Azevedo, Juurlink, Meenderinck, Terechko, Hoogerbrugge, Alvarez, Ramirez, et Valero. september 2009a. « A Highly Scalable Parallel Implementation of H.264 ». *Transactions on High-Performance Embedded Architectures and Compilers*, vol. 4, n° 2, p. 111-134.

Azevedo, Meenderinck, Juurlink, Terechko, Hoogerbrugge, Alvarez, et Ramirez. january 2009b. « Parallel H.264 Decoding on an Embedded Multicore Processor ». *Proceedings of the 4th International Conference on High Performance Embedded Architectures and Compilers*, p. 404-418.

Baik, Sihn, Kim, Bae, Han, et Song. december 2007. « Analysis and Parallelization of H.264 decoder on Cell Broadband Engine Architecture ». *IEEE International Symposium on Signal Processing and Information Technology*, p. 791-795.

Baker, Dalale, Chatha, et Vrudhula. 2009. « A scalable parallel H.264 decoder on the cell broadband engine architecture ». *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, p. 353-362.

Chen, Cao, Peng, Peng, Yu, et Zhang. april 2008. « A Memory-Efficient CAVLC Decoding Scheme for H.264/AVC ». *IEEE Conference on Advanced Communication Technology*, vol. 2, p. 1135-1138.

Chi, Juurlink, et Meenderinck. 2010. « Evaluation of parallel H.264 decoding strategies for the Cell Broadband Engine ». *Proceedings of the 24th ACM International Conference on Supercomputing*, p. 105-114.

Gurhanli, Chung-Ping Chen, et Hung. june 2011. « Coarse Grain Parallelization of H.264 Video Decoder and Memory Bottleneck in Multi-Core Architectures ». *International Journal of Computer Theory and Engineering*, vol. 3, n° 3, p. 375-381.

Hui et Hongpeng. october 2009. « Research of parallel decoding algrithm in H.264 on TILE64 ». *IEEE International Conference on Broadband Network and Multimedia Technology*, p. 500-503.

Iqbal et Henkel. april 2009. « Efficient constant-time entropy decoding for H.264 ». *IEEE Europe Conference and Exhibition on Design, Automation and Test*, p. 1440-1445.

Kim, Yoo, Shin, Choi, et Paik. august 2006. « Memory-Efficient H.264/AVC CAVLC for Fast Decoding ». *IEEE Transactions on Consumer Electronics*, vol. 52, n° 3, p. 943-952.

Lee, Park, et Chung. december 2008. « Optimization of RunBefore Decoder and First One Detector for MPEG-4 AVC/H.264 CAVLC Decoding ». *IEEE International Conference on Electronic Design*, p. 1-5.

Meenderinck, Azevedo, Juurlink, Mesa, et Ramirez. november 2009. « Parallel Scalability of Video Decoders ». *Proceedings on Image and Video Communications and Processing*, vol. 57, n° 2, p. 173-194.

Mesa, Ramírez, Azevedo, Meenderinck, Juurlink, et Valero. december 2009. « Scalability of Macroblock-level Parallelism for H.264 Decoding ». *2009 15th International Conference on Parallel and Distributed Systems*, p. 236-243.

Moon. june 2007. « A New Coeff-Token Decoding Method With Efficient Memory Access in H.264/AVC Video Coding Standard ». *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, n° 6, p. 729-736.

Moon. september 2008. « An Advanced Total_Zeros Decoding Method Based on New Memory Architecture in H.264/AVC CAVLC ». *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, n° 9, p. 1312-1317.

Moon, Kim, et Kim. september 2005. « An Efficient Decoding of CAVLC in H.264/AVC Video Coding Standard ». *IEEE Transactions on Consumers Electronics*, vol. 52, n° 3, p. 933-938.

Moon, Eom, et Ha. august 2008. « Efficient memory architecture for fast total_zeros decoding in H.264/AVC CAVLC decoder ». *IEEE International Conference on Multimedia and Expo*, p. 65-68.

Moriyoshi et Miura. august 2008. « Real-time H.264 Encoder with Deblocking Filter Parallelization ». *IEEE International Conference on Consumer Electronics*, p. 1-2.

Niconico. 2012. « Site web de la séquence vidéo Bad Apple ». <http://iitran.secchan.net/downloads/>.

Null et Lobur, 2006. *The Essentials of Computer Organization and Architecture*, p. 199-203. Jones and Bartlett Publishers, éd. 3rd.

Octasic. 2010. « Asynchronous Processor Design Evolution ». White Paper.

Qiu, Badawy, et Turney. november 2009. « An Architecture for Programmable Multi-core IP Accelerated Platform with an Advanced Application of H.264 Codec Implementation ». *Journal of Signal Processing Systems*, vol. 57, n° 2, p. 123-137.

Richardson, 2003. *H.264 and MPEG-4 Video Compression : Video Coding for Next-Generation Multimedia*, p. 159-222. John Wiley & Sons Ltd.

Schoffmann, Fauster, Lampl, et Boszormenyi, 2007. *An Evaluation of Parallelization Concepts for Baseline-Profile Compliant H.264/AVC Decoders*, p. 782-791. Springer Berlin / Heidelberg.

Seitner, Bleyer, Shreier, et Gelautz. 2008. « Evaluation of Data-Parallel Splitting Approaches for H.264 Decoding ». *Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia*, p. 40-49.

Seitner, Bleyer, Gelautz, et Beuschel. june 2011. « Evaluation of data-parallel H.264 decoding approaches for strongly resource-restricted architectures ». *Journal of Multimedia Tools and Applications*, vol. 53, n° 2, p. 431-457.

Sun, Wang, et Chen. 2007. « A Highly Efficient Parallel Algorithm for H.264 Encoder Based on Macro-Block Region Partition ». *Proceedings of the 3rd international conference on High Performance Computing and Communications*, p. 577-585.

Van Der Tol, Jaspers, et Gelderblom. 2003. « Mapping of H.264 decoding on a multiprocessor architecture ». *Proceedings of the SPIE on Image and Video Communications and Processing*, vol. 5022, p. 707-718.

Wenger, Hannuksela, Stockhammer, Westerlund, et Singer. February 2005. *RTP Payload Format for H.264 Video*.

Wiegand, Sullivan, Bjontegaard, et Luthra. July 2003. « Overview of the H.264/AVC video coding standard ». *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, n° 7, p. 560-576.

x264. 2012. « Site web officiel du logiciel x264 de VideoLAN Organization ». <http://www.videolan.org/developers/x264.html>.

Xiph.org. 2012. « Site web de Xiph.org Test Media ». <http://media.xiph.org/video/derf/>.